Universidad de las Ciencias Informáticas

Facultad 4

Videojuego Ace Fighter para la plataforma Cosmox

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Oscar Mastrapa Prats

Tutores:

MSc. Rubén Alcolea Núñez

Ing. Willian René Zaldivar Pupo

Ing. Yoandy Paz Perdigón

La Habana, julio de 2022

DECLARACIÓN DE AUTORÍA:

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

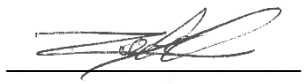
Para que así conste firmamos la presente a los 8 días del mes de julio del año 2022.

Oscar Mastrapa Prats

Autor

MSc. Rubén Alcolea Núñez

Tutor



Ing. Willian René Zaldivar Pupo

Tutor

Ing. Yoandy Paz Perdigón

Tutor

AGRADECIMIENTOS

No puedo expresar en palabras lo agradecido que estoy con mis padres por permitirme mantener y desarrollar mi pasión por los videojuegos.

Muchas gracias a mis tutores por la enorme ayuda ofrecida durante el desarrollo de este trabajo.

RESUMEN:

El desarrollo de videojuegos es una rama de la industria del entretenimiento que ha crecido enormemente desde su creación, brindando oportunidades de lucro y gran capacidad de comunicación. Esto convierte a los videojuegos en una alternativa atractiva para ofrecer entretenimiento acorde a los valores y principios de la sociedad cubana. En la Universidad de las Ciencias Informáticas se encuentra el Centro de Tecnologías Interactivas, el cual se encuentra enfocado en el desarrollo de la plataforma Cosmox. Esta ofrece un conjunto servicios que pueden ser usados por los desarrolladores en sus juegos para aumentar la cantidad de contenido y entretenimiento de sus productos. Uno de los servicios brindados por la plataforma es el de ranking. La inclusión de un ranking en videojuegos aporta valor de entretenimiento general ofrecido. Los videojuegos casuales que implementan un servicio en línea de ranking permiten a los jugadores compartir datos sobre el juego y competir contra otros jugadores. Como resultado se obtuvo un videojuego casual perteneciente al género Shoot' em up que implementa servicios de Cosmox. El proceso de desarrollo fue guiado por el Marco de trabajo ingenieril para el desarrollo de videojuegos, se utilizó C# como lenguaje de programación y el motor de videojuegos Unity.

Palabras clave: casual, shoot' em up, videojuego

ÍNDICE:

CONTENIDO

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El uso de las Tecnologías de Información y las Comunicaciones (TIC) en los últimos años ha permitido extender la informática a muchos sectores, ocupando cada vez más espacio en la vida y funcionamiento de la sociedad. Uno de los procesos que mayor impacto ha tenido en la sociedad es el desarrollo de videojuegos. Hoy en día, la industria del videojuego se ha consolidado como la principal contribuidora en la economía del entretenimiento y ocio, llegando a ser más relevante que industrias del mismo tipo como la música y el cine.

Los videojuegos son una vía de entretenimiento interactivo en el que uno o varios usuarios, mediante un dispositivo de entrada, se comunican con un sistema que posea imágenes de video. La plataforma en la que se desarrolle (o sistema) puede ser una computadora, consola, o incluso un celular. La interactividad usuario–sistema está dada por la capacidad del equipo de trabajo de planificar un sistema en el que el usuario se sienta cómodo y controle la situación [1].

A lo largo de la historia de los videojuegos, sus creadores han dado lugar a una variedad creciente de géneros en las distintas plataformas disponibles. Estos géneros se han conformado en torno a factores como: la representación gráfica, el tipo de interacción entre el jugador y la máquina, la ambientación, y su sistema de juego, siendo este último el criterio más habitual a tener en cuenta. Los géneros o categorías de videojuegos son una forma de clasificar los videojuegos en función fundamentalmente de su mecánica de juego. Sin embargo, existen otros factores como la estética o la temática que pueden influir también a la hora de definir ciertos géneros [2]. Es difícil que las personas no encuentren un videojuego capaz de satisfacer sus preferencias. Aquellos que no se consideran jugadores y solo consumen videojuegos durante periodos cortos de tiempo también encuentran su espacio dentro del mercado en los juegos conocidos como casuales. Los videojuegos casuales les proporcionan una forma de entretenimiento rápido y sencillo.

En la actualidad los videojuegos se han expandido a diferentes plataformas y categorías, produciendo un aumento en el mercado entorno a ellos. El carácter interactivo y su capacidad de comunicación aumentan el interés de las empresas por estos productos, trayendo como consecuencia un incremento en el número de videojuegos y dinero ingresado, así como el surgimiento de empresas especializadas en el sector.

La industria del videojuego gira en torno a pautas establecidas por el mercado y el desarrollo del videojuego. Las empresas que participan poseen roles esenciales a la hora de cumplir con el principal objetivo de vender

la mayor cantidad de productos. Los roles principales van desde la identificación de conceptos prometedores, hasta el desarrollo y comercialización, siempre asegurándose de que los productos sean atractivos para los usuarios. Las empresas buscan mantener una buena imagen para los usuarios ofreciendo productos con propuestas técnicas o estéticas interesantes. Involucrar a los usuarios con determinado videojuego sin siquiera estar listo mediante elaboradas campañas de marketing también es una práctica de la industria. Además, las empresas interesadas proporcionan financiamiento y soporte a aquellos proyectos prometedores, se encargan del estudio del mercado, ventas y distribución, así como del tratamiento de los datos generados a lo largo del proceso de desarrollo.

Una tendencia que ha ganado mucho espacio en la industria del videojuego es la posibilidad de interacción en línea. La interacción en línea y la inclusión de servicios o compras dentro de los videojuegos se vio revolucionada con el desarrollo y empleo del internet. Los videojuegos son puestos en servidores donde jugadores separados físicamente pueden interactuar y compartir la experiencia de juego, lo que aumenta enormemente el valor y potencial de la actividad de ocio ofrecida por el videojuego. Por otro lado, la oferta de servicios y compras en los juegos da la posibilidad a los jugadores de aumentar las formas que tiene de consumir y disfrutar de los videojuegos, además de que representan una fuente de ingresos extra o principal en algunos casos.

Los videojuegos son una opción atractiva para ofrecer entretenimiento acorde a los valores y principios de la sociedad cubana. Entre las instituciones del país que desarrollan videojuegos se encuentra el Centro de Tecnologías Interactivas (Vertex) de la Universidad de Ciencias Informáticas (UCI), con su línea de desarrollo de videojuegos. El centro se encuentra enfocado en el desarrollo de una plataforma de apoyo al desarrollo de videojuegos nombrada Cosmox. La plataforma brindará un conjunto de herramientas para desarrolladores terceros y jugadores, así como servicios de pasarela de pago, torneos y el hospedaje de partidas multijugador en línea por turnos y tiempo real.

Los desarrolladores cuentan con un sitio donde comunicar y analizar los videojuegos, así como interactuar con los jugadores. Uno de los servicios implementados en plataforma es el servicio de ranking. Este ha sido determinante en el éxito de algunos videojuegos como Coliseum [3], La Neurona 2 [4] y Kuba Kart [5]. La posibilidad de compartir las estadísticas del juego como datos, récord, tiempos, incentivan la afluencia de usuarios a la plataforma de videojuegos.

A nivel internacional han surgido varios videojuegos casuales que han tenido altos niveles de éxitos entre el público. Entre estos se encuentran Candy Crush [6] y SubWay Surfer [7], los cuales implementan servicios en línea similares a los antes mencionados. Parte del éxito de estos videojuegos se debe a la posibilidad de que los jugadores puedan publicar sus puntuaciones más altas, incentivando la competitividad entre los usuarios.

El desarrollo de videojuegos casuales en el centro Vertex no es nuevo. Las primeras propuestas surgieron como parte de maratones de desarrollo de videojuegos, tales como el Global Game Jam y el Pachamama Game Jam. Algunas de las propuestas más conocidas fueron la Súper Claria [8] y La Chivichana [9]. Estas propuestas no se beneficiaron de servicios como el ranking debido a la inexistencia de una plataforma que brindara este servicio. Se hace evidente que la presencia de la plataforma es indispensable para desarrollar propuestas de videojuegos atractivas para los usuarios y la implementación de servicios en línea puede contribuir a lograr este objetivo.

Luego del análisis realizado, se arriban las siguientes conclusiones:

- Los videojuegos casuales exitosos implementan sistemas de ranking en línea.
- Los videojuegos casuales desarrollados en el centro que no implementaron sistema de ranking fueron menos exitosos que aquellos que sí lo hicieron.
- Algunos videojuegos como Coliseum, Kuba Kart y la Neurona 2 tuvieron un aumento de sus ingresos a partir de su vinculación a los servicios de la plataforma.

Por la **problemática** antes expuesta, se propone el siguiente **problema a resolver**: ¿Cómo incorporar los servicios en línea de Cosmox en los videojuegos casuales?

El **objeto de estudio** es el desarrollo de videojuegos casuales, mientras que el **campo de acción** es el desarrollo de videojuegos Shoot’ em up.

Se define como **objetivo general**: Desarrollar un videojuego casual Shoot’ em up que implemente el servicio de ranking de la plataforma Cosmox.

Para darle cumplimiento al objetivo planteado se proponen las siguientes **tareas**:

1. Elaboración del marco teórico a partir del estado del arte actual referente al tema.

2. Análisis y selección de las herramientas y metodologías para el desarrollo de la propuesta de solución.
3. Diseño y conceptualización del videojuego como propuesta de solución.
4. Confección de la documentación necesaria para el desarrollo de la propuesta de solución.
5. Implementación del videojuego.
6. Implementación de pruebas.

El desarrollo de la investigación se apoyó en algunos **métodos científicos** que fueron muy útiles para guiar la investigación. A continuación, se explican los métodos de investigación que se utilizaron, los cuales se pueden clasificar en métodos teóricos y empíricos.

Métodos teóricos:

- El **método histórico-lógico** se utilizó para analizar la evolución a través del tiempo de los videojuegos.
- El **método analítico-sintético**: Empleado para analizar las interrelaciones entre los diferentes conceptos y definiciones relacionados con la investigación.
- El **método sistémico** se utilizó para determinar los componentes que forman parte del proceso de desarrollo de videojuegos y analizar la interacción que existen entre estos.

Métodos empíricos:

- La **observación** se empleó para estudiar distintos videojuegos. Estos sirvieron como objeto de análisis y comparación para establecer las características y elementos fundamentales que deben estar presente en un prototipo de videojuego del mismo tipo y género.

El presente trabajo se compone de tres capítulos, estructurados de la siguiente forma:

- **Capítulo 1 Fundamentación Teórica:** en este capítulo se definen los principales conceptos necesarios para el desarrollo de la investigación como lo son los videojuegos casuales. Se realiza un análisis de diferentes videojuegos para este fin, así como de las herramientas, tecnologías y métodos a usar.
- **Capítulo 2 Análisis y diseño de la solución:** en este capítulo se describe la propuesta de solución y los artefactos generados de acuerdo a la metodología seleccionada para el desarrollo de

videojuegos, estos son: documento de diseño del videojuego, especificación de mecanismos, descripción de requisitos no funcionales y arquitectura.

- **Capítulo 3 Implementación y pruebas de la solución:** en este capítulo se realiza una representación a nivel de componente de los mecanismos del videojuego. Se explica el estándar de codificación utilizado en su implementación y la implementación de métodos relevantes. Finalmente se presenta el resultado de las pruebas realizadas a la aplicación para validar su correcto funcionamiento y que cumpliese con el objetivo propuesto.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo, se describen los conceptos principales relacionados con los videojuegos con el objetivo de establecer un lenguaje común, que permita avanzar con el resto del trabajo. Se profundiza en el estudio de los videojuegos casuales y Shoot' em up. Se establecen las pautas a seguir en la planificación y desarrollo de un videojuego, haciendo énfasis en las tecnologías, herramientas y metodología a utilizar en el proceso de desarrollo del mismo.

1.1 Videojuego

Un videojuego es un software creado para entretenimiento, diversión e inmersión en un entorno virtual, que se fundamenta en la interacción entre una o varias personas y un dispositivo electrónico; estos elementos pueden ser un computador, una videoconsola o un teléfono móvil; por lo general, estos elementos se conocen como plataformas. De otra forma, se puede decir que es un software de apoyo al entretenimiento que, a partir de la interacción entre entornos virtuales, dispositivos electrónicos y usuarios, logra la simulación de una apariencia de la realidad deseada y la inmersión en ella de, por lo menos, un jugador [10].

El término juego se refiere a una actividad que es esencialmente libre/voluntaria, separada en el tiempo y el espacio, incierta e improductiva, que se rige por las reglas de la fantasía [11].

El juego incluye cualquier forma de software de entretenimiento por computador, usando cualquier plataforma electrónica, además consta de la participación de uno o varios jugadores en entorno físico o de red [12].

Clasificación y géneros de los videojuegos

Los videojuegos se pueden clasificar en géneros atendiendo a factores como el sistema de juego, el tipo de interactividad con el jugador o sus objetivos. Entre los géneros más comunes se encuentran los de aventura, acción, estrategia, agilidad mental, deportes, rol o simulación [13]:

- **Acción**: En este tipo de juegos las acciones básicas son mover el personaje y usar un arma. La mecánica del juego impone generalmente al jugador tener buenos reflejos y precisión. Algunos ejemplos son: DOOM, Half Life, Quake.

- **Deporte**: Los juegos de deporte son aquellos que simulan juegos de deporte real, entre ellos se encuentran: Golf, Tony Hawks, FIFA.
- **Aventura**: Videojuegos orientados al descubrimiento de una trama narrativa con tendencia a la jugabilidad rígida. Se encuentra dentro de este género el juego Colossal Cave Adventure.
- **Rol**: Videojuegos orientados a la reconstrucción de una narrativa con jugabilidad abierta. Se caracteriza por la adopción un rol y la búsqueda de puntos de experiencia, que se obtienen por diversas acciones meritorias en el juego y que sirven para mejorar progresivamente las habilidades del personaje. Presentan una estructura de misiones diseminadas en el juego, entre las que el jugador tiene un cierto margen de libertad en su elección y en el orden de abordarlas. Algunos videojuegos del tipo de rol son: The Elder Scrolls IV: Oblivion, Dark Souls.
- **Agilidad mental**: Estos son juegos donde el jugador debe pensar y agilizar la mente. El objetivo es resolver ejercicios con dificultad progresiva para desarrollar la habilidad mental. Algunos videojuegos de este género son: Brain Academy, La Neurona.
- **Simulación**: Pretenden emular ante los jugadores vivencias y situaciones lo más realistas posible. La diferencia entre un juego de simulación y otro radica en el nivel de detalle del entorno y el control que puedan ejercer sobre el mismo. De este tipo de videojuego se destacan títulos como Eurotruck.
- **Estrategia**: Requieren que el jugador ponga en práctica sus habilidades de planeamiento y pensamiento para maniobrar, gestionando recursos de diversos tipos como materiales, humanos o militares para conseguir la victoria. Dentro de este género se encuentran: Star Craft, Age of Empires.

Hay que mencionar que existe una clasificación o tipología para cierto grupo de videojuegos que son denominados videojuegos casuales. Estos pueden tener elementos de los géneros antes expuestos, pero al poseer ciertas características como la complejidad del argumento o jugabilidad, que los hace merecedores de tal clasificación. En estos últimos se enfocará la investigación con el fin de dar solución al objetivo general planteado.

1.2 Videojuego Casual

Los videojuegos casuales carecen de argumentos y mecánicas complicadas, en su lugar optan por jugabilidad simple y entretenida [14]. Al no contar con gran inmersión o la necesidad de mucha concentración se vuelven la opción ideal para pasar periodos de tiempo cortos, aumentando la cantidad de veces que los jugadores recurren a estos.

Estos pueden tener cualquier tipo de mecánica de juego y clasificarse dentro de cualquier otro género. Son típicamente distinguibles por sus reglas simples y que no requieren excesivo compromiso en contraste con la mayoría de juegos, más complejos. Los videojuegos casuales están dirigidos a un público principal de jugadores casuales [14]. Un jugador de videojuegos casual es un tipo de jugador de videojuegos cuyo interés o tiempo dedicado a jugar videojuegos es más reducido en comparación a otras actividades. Se puede catalogar como jugador casual a todos aquellos que muestran apenas un interés pasajero en los videojuegos, por lo que es difícil categorizarlos como un grupo. Por esta razón, los juegos que tratan de atraer al jugador casual tienden a implementar reglas simples y facilidad de juego, con el objetivo de presentar una experiencia de coger y jugar (pick up and play) que la gente de casi cualquier edad o nivel de habilidad pueda disfrutar [15].

No existe definición precisa a la hora de clasificar los videojuegos casuales. De forma parecida a la clasificación general de los videojuegos, estos pueden ser agrupados según las características que tienen y modo de juego, pero es más común encontrarlos de los siguientes géneros:

- Puzzles (Where's My Water, Cut the Rope)
- Arcade (Geometry Dash, Metal Slug)
- Acción (Meltdown, Nuclear Throne)
- Estrategia (Dinner Dash, Clash of Clans)
- Aventura (Minecraft, Terraria)
- Shoot'em up (Space Invaders, SpaceWar!, Chicken Invaders)

1.3 Videojuegos Shoot 'em up

Existe cierta confusión en los orígenes del género. Algunos defienden que el primer videojuego Shoot' em up fue el SpaceWar!, sin embargo, el videojuego Space Invaders fue el que lo popularizó y, por tanto, la mayoría le atribuye a este la creación del género.

El género de los Shoot' em up es un subgénero de los shooters (disparos) donde el jugador controla un único personaje o vehículo, a menudo una nave espacial u otro tipo de vehículo volador, y debe eliminar a multitud enemigos que aparecen en pantalla de manera simultánea al tiempo que esquiva disparos [16].

El avatar del jugador es típicamente un vehículo que se encuentra bajo constante ataque. Por lo tanto, el objetivo del jugador es disparar lo más rápidamente posible a todo lo que se mueva o lo amenace [17].

Las principales habilidades requeridas en un Shoot' em up son las capacidades de reacción rápida y de memorizar los patrones de ataque enemigo. Algunos juegos presentan cantidades enormes de proyectiles enemigos, y el jugador debe memorizar sus patrones de una a otra partida si quiere sobrevivir [18].

Algunos de los títulos más populares del género son Space Invaders, Star Fox, Metal Slug, Raiden V, Velocity 2X, Beat Hazard 2, Cuphead [19].

Tipos de Shoot' em up

Los Shoot' em up se categorizan por sus elementos de diseño, especialmente el punto de vista del jugador y el movimiento [20]:

- Fixed Shooter: constan de niveles cada uno de los cuales cabe en una sola pantalla. El movimiento del protagonista está fijado a un único eje, y los enemigos atacan en una única dirección [21].
- Rail Shooter: limitan al jugador a moverse por la pantalla mientras el juego sigue una ruta específica [22].
- Tube Shooter: presentan una nave que vuela por un tubo abstracto [23].
- Scrolling Shooter: se incluyen juegos con desplazamiento vertical u horizontal de la pantalla. En un Shoot 'em up con desplazamiento vertical, la acción se ve desde arriba y la pantalla se desplaza de arriba hacia abajo. Ello tiene la ventaja de permitir complejos patrones de enemigos, así como de hacer posible que gráficos simples, incluso, funcionen de forma convincente [20].
- Bullet Hell: es un Shoot 'em up en el cual toda la pantalla está a menudo casi completamente llena de balas enemigas.
- Run and gun: describe un tipo de Shoot 'em up en el que el protagonista se desplaza a pie, a veces con la posibilidad de saltar. Estos juegos usan perspectivas de desplazamiento lateral, vertical o isométrica, y pueden presentar además movimiento multidireccional [20].

Debido a las posibilidades que brindan las pantallas de los dispositivos Android, se decide centrar la investigación en los videojuegos Shoot' em up con características de los Scrolling Shooter y Bullet Hell.

Sistemas homólogos

Space Invaders: es imposible hablar del género Shoot' em up sin hablar de este clásico. Durante el juego se puede apreciar que el jugador se mueve a los lados mientras dispara. Los enemigos aparecen en

grandes formaciones o grupos. Se desplazan de un lado a otro mientras avanzan desde la parte superior de la pantalla hasta la inferior. Estos ocasionalmente disparan un proyectil que avanza en línea recta hacia abajo. Si impacta con el jugador, este pierde una vida. El jugador tiene un número limitado de vidas. Eliminar enemigos le concede puntos. Varios jugadores pueden competir por ver quien alcanza la mejor puntuación [24]. Antaño esto se realizaba en la misma consola o máquina arcade, hoy en día es posible mediante el empleo de servicios por internet.

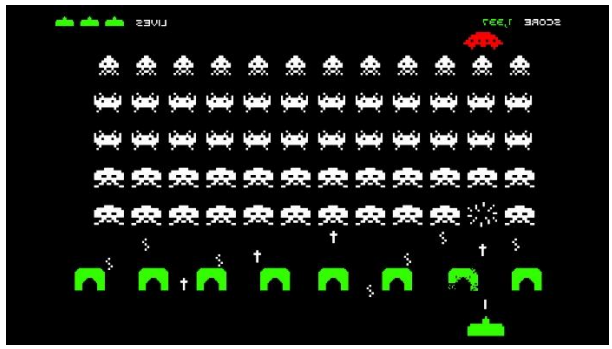


Figura 1. Videojuego Space Invaders.

Radiant: es un videojuego para plataformas móviles que toma inspiración directa de Space Invaders y añade nuevos elementos. El jugador mueve su nave de un lado al otro, y dispara de forma automática. El jugador puede equipar diferentes tipos de armas, las cuales puede mejorar para hacer más daño o atacar más rápido. Existen diferentes tipos de enemigos, los cuales tienen diferentes tipos de debilidades y fortalezas, lo que obliga al jugador a plantearse una estrategia a la hora de seleccionar el arma que usará. El jugador tiene vidas limitadas y puede resistir algunos impactos antes de ser destruido. Algunos enemigos apuntarán y dispararán hacia el jugador, los cuales aparecerán gradualmente uno detrás del otro en distintas trayectorias según el nivel hasta conformar un grupo en la pantalla, al ser derrotados concederán puntos y dinero de forma aleatoria. Durante el juego el jugador puede mejorar sus armas o comprar vidas con el dinero. Para variar el contenido el videojuego tiene una serie de niveles donde asteroides avanzan desde la parte superior de la pantalla hacia la inferior, en los que el jugador debe evitar colisionar con ellos [25]. Durante el desarrollo del videojuego los jugadores pueden competir por ver quien alcanza la mejor puntuación.

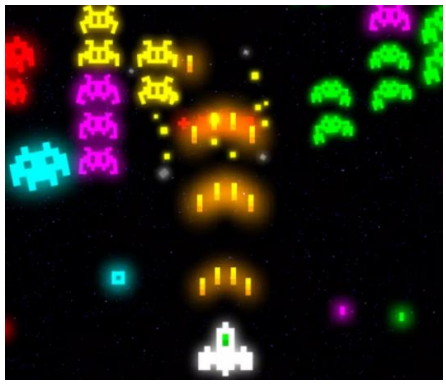


Figura 2. Videojuego Radiant.

Beat Hazard: es un videojuego cuya dificultad y jugabilidad varía en función de la música que se reproduzca. El jugador es libre de moverse por toda la pantalla, cuenta con un número limitado de vidas y debe de enfrentarse a numerosos enemigos que son generados según el ritmo de la música. Los enemigos junto con sus disparos a menudo inundan la pantalla de obstáculos para el jugador. El jugador puede usar habilidades que le conceden ventajas adicionales. La nave del jugador se puede mejorar. El videojuego tiene modos de juegos infinitos en los que los jugadores pueden competir por ver quién alcanza la mejor puntuación en las tablas de clasificación en línea [26].

Figura 3. Videojuego Beat Hazard.

Chicken Invaders: este juego también permite la movilidad del jugador por toda la pantalla. El fondo se desplaza verticalmente hacia abajo dando la impresión de que el jugador siempre se mueva hacia el frente. Los enemigos describen trayectorias definidas distintas en cada nivel y al ser destruidos proporcionan puntos y objetos. El jugador puede usar tres tipos de armas con patrones de ataque distintos. Puede usar una habilidad para limpiar la pantalla de enemigos. Tiene vidas limitadas, pero puede conseguir más

acumulando puntos. Los jugadores pueden compartir las puntuaciones alcanzadas en tablas de posiciones en línea [27].

Figura 4. Videojuego Chicken Invaders.

A continuación, se muestra un resumen de las características de interés para la propuesta presentes en los sistemas homólogos estudiados:

Tabla 1. Características de los sistemas homólogos.

Videojuego	Movimiento del jugador	Varios tipos enemigos	Varios tipos de armas	Se puede usar habilidades	Posibilidad de mejorar características	Generación procedural de contenido	Servicios en línea
Space Invaders	horizontal	sí	no	no	no	no	no
Radiant	horizontal	sí	sí	sí	sí	no	sí
Beat Hazard	libre	sí	no	sí	sí	sí	sí
Chicken Invaders	libre	sí	sí	sí	no	no	sí

Homólogos nacionales

En el mercado nacional se puede encontrar ejemplos de videojuegos juegos Shoot' em up para las plataformas móviles, dichos ejemplos son:

- SpaceDefender.
- Ship vs Rocks.
- Batalla de Papel.

Para el análisis de estos ejemplos se consideraron como características importantes: la forma de adquisición del videojuego, refiriéndose a si este es de pago o libre de costo, ya que es un factor influyente a la hora del usuario decidir si consumir o no el producto. También se tuvieron en cuenta los gráficos, ya que estos son responsables de atraer visualmente a los jugadores. Como opinión del público se interpretó la puntuación que le atribuyen los consumidores a los productos en la plataforma Apklis.

Tabla 2. Homólogos nacionales.

Juego	Autor	Es de Pago	Gráficos	Opinión del público	Observación
SpaceDefender	Rafael Benito Pérez Labañino	sí	sprites 2d	Posee una media de 3.3 en Apklis	Pocas descargas
Ship vs Rocks.	Rafael Benito Pérez Labañino	sí	sprites 2d	Ha sido comprado, pero no posee ninguna opinión aún	Pocas descargas
Batalla de Papel.	Mayelin Pupo Santos	sí	sprites 2d	Posee una media de 3.3 en Apklis	La versión demo gratis posee muchas más descargas.

De forma general los juegos son para un jugador y por los recursos proporcionados por los autores en la plataforma Apklis, poseen características de los juegos Shoot' em up, sin embargo, ninguno posee funcionalidades que permitan a los jugadores competir en un ranking nacional o salvar partidas independientemente del dispositivo donde se juega. Dichas funciones agregarían valor a la propuesta del

juego. Después del estudio y análisis de los videojuegos expuestos anteriormente, se identifican como partes fundamentales de la propuesta de solución los siguientes elementos:

- Movilidad del jugador: este debe ser capaz de moverse para apuntar y esquivar. El movimiento propuesto por juegos como Beat Hazard y Chicken Invaders resulta más atractivo. El jugador cuenta con más espacio para maniobrar y planear su ruta. Este tipo de movimiento permite un buen aprovechamiento del tamaño de las pantallas de los dispositivos con sistemas Android.
- Disparos: definidos por las armas, estos tienen patrones y distintos atributos como velocidad y daño. Enriquecen la jugabilidad dando la oportunidad al jugador de elegir cuál arma o tipo de disparos quiere usar.
- Habilidades: proporcionan al jugador elementos extra de interacción con los enemigos.
- Enemigos: elemento indispensable en este tipo de videojuegos. El comportamiento de estos suele estar definido por cada nivel o tipo de enemigo. Para este ejemplo resulta más atrayente la alternativa de Beat Hazard, el cual genera los enemigos en dependencia de la música. Para este método se usa la Generación Procedural de Contenido (GPC). La GPC se emplea para crear contenido automáticamente usando algoritmos predefinidos y esta se ha convertido en un foco importante de investigación en los últimos años. Esta técnica tiene la viabilidad para reducir esfuerzos manuales del diseño de cada parte del juego y el tiempo requerido para el desarrollo del mismo. La GPC brinda soporte a los diseñadores para desarrollar el contenido con costo y esfuerzo reducido. También puede ayudar a los jugadores a disfrutar del contenido dinámico dado que la utilización de GPC para la creación automática de historias de juego, provee un espacio de juego complejo y diverso [28]. El empleo de métodos de GPC puede añadir variación y dinamismo a las partidas.
- Movimiento del escenario: esta característica proporciona la perspectiva al jugador de que avanza por el espacio y no se encuentra fijo en una zona del juego.
- Capacidad de mejora: es un elemento que mantiene incentivado al jugador, mejorar los atributos como el daño a la cantidad de vida que se tiene permite mejorar el rendimiento durante el juego, lo que pude traducirse en satisfacción para el jugador.
- Servicios en línea: estos elementos añaden valor a la propuesta de entretenimiento del videojuego. Permitir a los jugadores competir por alcanzar la mejor puntuación es común en los juegos Shoot'

em up, ya que estos normalmente son para un solo jugador en los que se solía implementar rankings locales, pero el uso de internet permite la creación de tablas de posiciones regionales o globales, esto permite la participación de muchos más jugadores y un aumento de la competitividad. Esta característica puede ser empleada a través de los servicios ofertados en el portal Cosmox.

1.4 Cosmox

El portal de videojuegos Cosmox tiene como objetivo ofrecer un espacio virtual, que sirva como plataforma integradora de todo el conjunto de contenidos y servicios afines a los videojuegos cubanos, fundamentalmente desarrollados por la alianza estratégica entre la Universidad de las Ciencias Informáticas y los Estudios de Animación ICAIC [29].

El portal pretende crear un canal de comunicación interactivo, dinámico y comprometido con los contenidos cubanos. Adicionalmente a los enlaces de descarga de los videojuegos para diferentes plataformas, Cosmox permite a los usuarios interactuar con el portal desde los propios videojuegos, con el objetivo de publicar o compartir con la comunidad el avance dentro de los juegos.

Por otra parte, el portal de videojuegos es la interfaz inicial para el usuario de una plataforma más amplia, enfocada en el desarrollo de servicios para videojuegos en línea. El objetivo final es que los jugadores puedan competir contra otros en diferentes lugares del país. También se trabaja en la adición de servicios de salvas, que permitan conservar el progreso del usuario cuando se cambia de dispositivo [30]. El estudio de la plataforma permitió la identificación de los siguientes servicios:

- Sección de Videojuegos: contiene una descripción detallada de los videojuegos, así como los enlaces de descarga para las diferentes plataformas. De forma adicional algunos videojuegos ofrecen la posibilidad de descarga de la música, el póster y los videos promocionales.
- Rankings: muestra la tabla de posiciones de los mejores jugadores por videojuegos o niveles de videojuegos.
- Noticias: contiene información de los temas más novedosos referente a los videojuegos en el ámbito nacional e internacional.
- Historial: muestra en orden cronológico los lanzamientos de los videojuegos.
- FAQ: facilita un conjunto de preguntas frecuentes y sus respuestas, con el objetivo de esclarecer a los usuarios en el uso del portal y sus servicios.

- Contacto: posibilita las vías de comunicación de los usuarios con los administradores del portal.

1.5 Metodología de desarrollo de software

Para el desarrollo de los videojuegos, Vertex utiliza el Marco de trabajo ingenieril para el proceso de desarrollo de videojuegos. El mismo se centra en las mecánicas que definen el comportamiento en un videojuego, lo que posibilita una adecuada comprensión por parte del equipo de análisis y desarrollo del atributo de calidad y jugabilidad. De igual forma se acoge a un modelo de desarrollo incremental para garantizar la retroalimentación y entrega gradual del videojuego. Se compone por cinco etapas que complementan el proceso de desarrollo de los videojuegos: Conceptualización, Diseño, Implementación, Prueba y Mantenimiento [10].

A continuación, se define la estructura de la metodología:

- Etapa 1: Conceptualización
    - Definir el género sobre el cual se desarrollará el videojuego.
    - Describir la mecánica del videojuego.
    - Especificar las metas para la experiencia del jugador.
- Etapa 2: Diseño
    - Describir los elementos formales que definen la estructura del videojuego.
    - Describir los elementos dramáticos que definen el entretenimiento del videojuego.
    - Diseñar las pantallas gráficas elementales que forman la estructura del videojuego.
    - Describir los elementos dinámicos que definen las mecánicas o mecanismos del videojuego.
    - Validar los mecanismos teniendo en cuenta criterios técnicos para su implementación.
    - Modelar el diagrama de paquetes de mecanismos teniendo en cuenta la distribución arquitectónica.
    - Describir la concepción de los mecanismos sobre la distribución arquitectónica diseñada.
    - Modelar el comportamiento de los mecanismos mediante diagramas de transición de estado.
    - Mantener una trazabilidad bidireccional entre cada elemento del videojuego.
    - Describir las características no funcionales del videojuego.
- Etapa 3: Implementación
    - Diseñar los componentes que encapsulan la implementación.

    - Desarrollar las mecánicas especificadas y diseñadas.
- Etapa 4: Prueba
    - Desarrollar pruebas Alpha.
    - Desarrollar pruebas Beta.
    - Registrar defectos durante las pruebas realizadas.
- Etapa 5: Mantenimiento
    - Realizar análisis Postmortem.
    - Retomar la etapa de Diseño.

1.6 Herramientas y tecnologías

Lenguaje de modelado UML

Lenguaje Unificado de Modelado (UML) es un lenguaje de propósito general que ayuda a especificar, visualizar y documentar modelos de sistemas software, incluido su estructura y diseño, de tal forma que se unifiquen todos sus requerimientos [31]. El objetivo principal del UML es estandarizar el modelado de sistemas software. UML proporciona el vocabulario y reglas necesarias para combinar y construir representaciones, modelos conceptuales y físicos del sistema y permite representar varios modelos, compartiendo características específicas de cada uno como los nombres de clases, atributos y métodos.

Herramienta de modelado

Para el modelado de la solución de utilizará Visual Paradigm en su versión 8.0, ya que es una de las herramientas consideradas como completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. La herramienta cuenta con funciones que permiten generar código a partir de la representación de los modelos y viceversa [32].

Motor de videojuego Unity

Unity es lo que se conoce como un motor de desarrollo o motor de juegos. El término motor de videojuego (*game engine*) hace referencia a un software, el cual tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo, en este caso, de un videojuego [33].

Dentro de las funcionalidades típicas que tiene un motor de videojuegos se encuentran las siguientes:

- Motor gráfico para dibujar gráficos 2D y 3D.
- Motor físico que simule las leyes de la física.
- Animaciones.
- Iluminación.
- Sonidos.
- Inteligencia Artificial.
- Programación o scripting.

Herramienta de modelado 3D Blender

Con Blender se pueden crear visualizaciones 3D, así como imágenes fijas, animaciones 3D, tomas VFX y edición de video. La herramienta se adapta bien a las personas y los estudios pequeños que se benefician de su canalización unificada y su proceso de desarrollo receptivo. Al ser una aplicación multiplataforma, Blender se ejecuta en sistemas Linux, Mac, así como en Windows. También tiene requisitos de memoria y espacio en disco relativamente pequeños en comparación con otras herramientas de creación 3D. Su interfaz emplea OpenGL para proporcionar una experiencia consistente en todo el hardware y plataformas compatibles [34].

Herramienta de edición de imagen Gimp

GIMP es un editor de imágenes multiplataforma disponible para GNU/Linux, Mac, Windows y más sistemas operativos. Es software libre, por lo que se puede cambiar el código fuente y distribuir los cambios. GIMP brinda herramientas sofisticadas para realizar diversos trabajos, estos incluyen el diseño gráfico, la fotografía e ilustración. GIMP permite el incremento de las funciones que puede realizar o el mejoramiento de las existentes gracias a las muchas opciones de personalización y complementos de terceros existentes en la comunidad [35]. Muchas pequeñas empresas lo utilizan para crear logotipos o gráficos de forma gratuita. Algunas de sus prestaciones no tienen nada que envidiar a otras licencias comerciales. De hecho, se ha convertido en alternativa a Photoshop en algunos casos. La primera versión del programa se ideó para GNU/Linux. En la actualidad, también existen adaptaciones propias para Windows y Mac OS X. Este programa de software libre soporta la mayoría de ficheros gráficos [36].

Lenguaje de programación

C# (C sharp) es un lenguaje de programación orientado a objetos y componentes, tiene una sintaxis muy similar a Java, posee una librería de clases muy completa y bien diseñada. Permite mantener múltiples versiones de clases en forma binaria, colocándolas en diferentes espacios de nombres, esto permite que versiones nuevas y anteriores de software puedan ejecutarse de forma simultánea [37]. C# es un lenguaje de programación moderno, orientado a objetos y con seguridad de tipos. Permite a los desarrolladores crear muchos tipos de aplicaciones sólidas y seguras que se ejecutan en .NET. Tiene sus raíces en la familia de lenguajes C y resulta inmediatamente familiar para los programadores de C, C++, Java y JavaScript [38].

Entre sus principales características se encuentran: la capacidad para desarrollar componentes de software que se puedan usar en ambientes distribuidos, la portabilidad del código fuente y la construcción aplicaciones económicas en cuanto a memoria y procesado [37].

Editor de texto VS Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y Mac, es gratuito y de código abierto. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias [39]. Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta como una aplicación de escritorio. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity) [40].

Conclusiones parciales del capítulo

A partir del análisis del estado del arte actual sobre el tema de investigación propuesto, es posible arribar a las siguientes conclusiones:

- El análisis de los conceptos y aspectos fundamentales de los videojuegos casuales y del género Shoot’ em up, proporcionaron un mejor entendimiento de los mismos, permitiendo llegar a la propuesta de solución.
- Se plantea el desarrollo de un videojuego Shoot’ em up que permita el movimiento del jugador por la pantalla, tenga varias armas y habilidades, exista la posibilidad de mejorar características, genere

formaciones de enemigos de forma procedural y permita a los usuarios publicar sus puntuaciones para competir entre ellos a través del servicio de ranking de Cosmox.

- El empleo el Marco de trabajo ingenieril para el proceso de desarrollo de videojuegos permitirá enfocar los artefactos necesarios a utilizar, proporcionando así una mayor descripción y organización de los requerimientos del software, reduciendo la curva de aprendizaje y el tiempo de desarrollo.

CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

En el siguiente capítulo se da a conocer la descripción de la propuesta de solución. Los procesos y actividades descritas se realizan según lo propuesto por el Marco de trabajo ingenieril para el proceso de desarrollo de videojuegos. Se realiza la definición de los elementos formales y dramáticos, la especificación de los mecanismos, la descripción de los patrones de diseño aplicados y la representación de los diagramas de clases del diseño.

2.1 Conceptualización de la propuesta de solución

Se propone como solución el videojuego Ace Fighter, el cual se concibe como un videojuego casual para plataformas móviles, por lo tanto, posee un argumento y jugabilidad sencillos. Como el nombre sugiere, el jugador controla un caza espacial durante el desarrollo de un conflicto bélico en el espacio. El jugador debe hacer frente a una fuerza superior en número de cazas enemigos durante todo el tiempo que pueda, ya que el escenario de juego es infinito. El jugador tendrá que esquivar los disparos enemigos y acertar los suyos para derribarlos. Una vez derribado el caza del jugador, este obtendrá un resumen de lo logrado durante la partida. Al ser un juego para dispositivos con sistema Android, se desea que la visión de la pantalla se obstruya lo menos posible por lo que el jugador solo debe controlar la posición de su caza con una mano, ya que este estará programado para disparar por su cuenta.

Jugabilidad

El jugador tendrá la opción de seleccionar el arma y la munición que usará en la partida en el menú principal del juego. Estas decisiones afectarán el desempeño de la partida y aportarán variación al desarrollo de esta. El jugador también tiene la posibilidad de hacer uso del servicio de ranking de Cosmox, para lo cual deberá identificarse en la sección del perfil. Además, puede acceder a la sección de ayuda para obtener información útil sobre el juego y consultar los autores de los recursos utilizados en la sección de créditos. El jugador podrá alterar su desempeño durante el juego gracias a las opciones ofrecidas en las secciones del hangar, el cual funciona a modo de tienda y la armería.

Al ser un videojuego perteneciente al género Vertical Scrolling de los Shoot' em up, la perspectiva será desde la parte superior (*Top View*). El jugador podrá desplazar el caza en cuatro direcciones: izquierda, derecha, adelante y atrás. Este posicionamiento de la vista de juego permitirá al jugador observar con precisión la posición de su caza, los enemigos y los proyectiles. Existirán dos tipos de enemigos con

variación en atributos de vida y puntuación. El jugador contará con tres tipos de armas comunes al igual que los enemigos; y estos últimos poseen además una especial, la cual apuntará en todo momento al jugador para obligarle a esquivar. El jugador podrá tener a su disposición una serie de habilidades: escudo, misiles y reparación, que puede activar durante la partida para mejorar su rendimiento. Dichas habilidades son limitadas y deben de reponerse en el hangar donde a su vez, el jugador podrá comprar habilidades y mejorar las estadísticas de su caza. Entre las mismas se pueden encontrar: puntos de vida, daño de las armas y tiempo entre ráfagas de disparos. La puntuación de la partida estará definida por la cantidad de tiempo que sea capaz de sobrevivir el jugador más los puntos obtenidos por derribar cazas enemigos. Los créditos son la moneda del juego para comprar mejoras y habilidades, estos se adquieren en determinadas cantidades según la puntuación obtenida en la partida.

Metas para la experiencia del jugador:

- El jugador recurrirá a la experiencia del juego como método rápido y sencillo de entretenimiento.
- El jugador será incentivado a ver cuánto tiempo puede sobrevivir.
- El jugador estará incentivado a mejorar su puntuación personal en cada partida.
- El jugador querrá mejorar las estadísticas de su caza para mejorar su desempeño durante la partida.
- El jugador querrá mejorar su puntuación en el ranking de puntos.

2.2 Diseño del Videojuego

A continuación, se definirán los elementos principales que conforman el juego: los elementos formales, los elementos dramáticos y prototipos de las pantallas gráficas, así como la descripción de los mecanismos y los artefactos que definen la arquitectura del software.

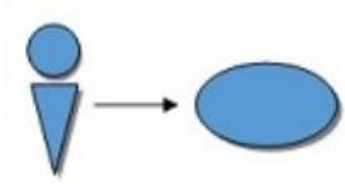
Elementos formales

Los elementos formales definen las características, objetos y comportamientos del juego.

Jugadores

Tabla 3. Jugador.

Aspecto	**Descripción**
Invitación a jugar	Botón de comienzo “Empezar”
Cantidad	Un solo jugador

Roles	Es quien interactúa con el videojuego
Patrón de interacción	Individual vs Juego 

Objetivos

Tabla 4. Objetivos.

Objetivos	**Descripción**
Sobrevivir la mayor cantidad de tiempo posible y derribar la mayor cantidad de enemigos.	El cumplir con este objetivo permite al jugador obtener mayor cantidad de puntos y créditos al finalizar la partida.
Mejorar las estadísticas del caza.	El cumplir con este objetivo aumentará paulatinamente el rendimiento del jugador en la partida.
Obtener la mayor cantidad de puntos en la partida.	El cumplir con este objetivo mantiene al jugador motivado y esforzándose más por superarse para alcanzar mejores lugares en el ranking de puntos.

Procedimientos

- En el menú principal el jugador puede seleccionar una de tres armas para jugar.
- En el menú principal el jugador puede seleccionar uno de dos tipos de munición para jugar.
- El jugador desplaza su caza por la pantalla.
    - Debe esquivar los disparos enemigos.
    - Debe hacer que los proyectiles que dispare su caza impacten a los enemigos.
    - Puede activar las habilidades que tiene disponible.
- El jugador puede adquirir mejoras permanentes y habilidades en el hangar.
- Al finalizar la partida, el jugador puede elegir entre jugar o reiniciar la partida, ir al hangar o salir al menú principal.

- Al pausar la partida, el jugador puede elegir entre continuar, reiniciar la partida o salir al menú principal.
- En el hangar el jugador puede comprar mejoras permanentes y habilidades mientras tenga créditos disponibles.

Reglas

- Jugador
    - El jugador comienza con puntuación de cero puntos.
    - El caza del jugador comienza con 100 puntos de vida.
- Recursos consumibles
    - El jugador comienza con cero créditos.
    - El jugador comienza con cero usos de la habilidad Reparación y puede acumular un número indefinido de esta habilidad.
    - El jugador comienza con cero usos de la habilidad Escudos y puede acumular un número indefinido de esta habilidad.
    - El jugador comienza con cero usos de la habilidad Misiles y puede acumular un número indefinido de esta habilidad.
- Armas
    - El jugador solo puede equipar un tipo de arma.
    - El jugador solo puede equipar un tipo de munición.
    - Desde la sección Armería, el jugador puede cambiar el arma y la munición que usará durante la partida.
    - Propiedades iniciales del arma Cañón.
        - Daño de 20.
        - Ráfagas de tres proyectiles.
        - Frecuencia de disparo entre ráfagas 1.0 segundos.
    - Propiedades iniciales del arma Artillería.
        - Daño 50.
        - Ráfagas de un proyectil.
        - Frecuencia de disparo entre ráfagas 1.5 segundos.

- Propiedades iniciales del arma Minigun.
    - Daño 10.
    - Ráfagas de 5 proyectiles.
    - Frecuencia de disparo entre ráfagas 1.5 segundos.
- Munición
    - Munición con alto poder explosivo (HE) hace 20 por ciento más daño a los enemigos ligeros, pero 20 por ciento menos daño a los pesados.
    - Munición con gran capacidad para atravesar armadura (AP), hace 20 por ciento más daño a los enemigos pesados, pero 20 por ciento menos daño a los ligeros.
- Mejoras de armas (valor en que mejora y precio)
    - Las mejoras afectan a todas las armas del jugador, estén activas o no.
    - La mejora “Mejorar munición” aumentará el daño de los proyectiles del jugador permanentemente en 2.
        - Costo 100 créditos, cada compra aumenta el costo en 10.
    - La mejora “Cargador automático” reduce de forma permanente el valor de la frecuencia de disparo entre ráfagas en 0.1 segundos, al mejorarse 10 veces, el valor se restablece en el inicial, pero añade otro proyectil a la ráfaga.
        - Costo 100 créditos, cada compra aumenta el costo en 10.
    - La mejora “Reforzar casco” aumenta la cantidad de puntos de vida del caza del jugador de forma permanente en 100.
        - Costo 100 créditos, cada compra aumenta el costo en 10.
- Atributos de los enemigos según el tipo
    - Ligeros, son pequeños y rápidos, van armados con dos cañones y una torreta automática.
        - 100 puntos de vida iniciales.
        - Aumentan la cantidad de vida que tienen en 50 cada 30 segundos.
        - El daño de las armas aumenta en 10 cada 30 segundos.
    - Pesados, son grandes y lentos, armados con dos cañones, dos armas de artillería y dos torretas automáticas.
        - 200 puntos de vida iniciales.

        - Aumentan la cantidad de vida que tienen en 50 cada 30 segundos.
        - El daño de las armas aumenta en 10 cada 30 segundos.
- Durante la partida
    - Cada segundo de partida proporciona 10 puntos.
    - Los enemigos son destruidos si sus puntos de vida llegan a 0.
    - Cada enemigo ligero destruido proporciona 100 puntos.
    - Cada enemigo pesado destruido proporciona 200 puntos.
    - Si los puntos de vida del caza del jugador llegan a 0 termina la partida.
    - Los enemigos aparecen en oleadas cada 12s en formaciones generadas proceduralmente.
    - Las armas de los enemigos, Cañón, Artillería y Minigun, poseen los mismos valores en sus propiedades que las iniciales del jugador.
    - La dificultad escala según el tiempo de partida, cada 30 segundos aumenta el daño de los enemigos en 10 y sus puntos de vida en 50.
    - Si la puntuación de la partida fue mayor que la mejor puntuación del jugador, esta pasa a tomar el valor de la puntuación de la partida.
    - Cada 100 puntos se le proporciona al jugador 10 créditos.

Recursos

- Vida: El jugador y los enemigos cuentan con una cantidad de vida limitada durante la partida.
- Créditos: moneda de intercambio por mejoras y habilidades.
- Reparación: habilidad que restaura 100 puntos de vida al caza del jugador.
    - Costo 100 créditos.
- Escudo: esta habilidad proporciona al jugador una barrera frontal que lo protege del daño durante 10 segundos.
    - Costo 100 créditos.
- Misiles: habilidad lanza desde atrás del jugador un barrido de misiles para limpiar la pantalla de enemigos.
    - Costo 100 créditos.
- Caza del jugador: representa el protagonista del juego, es controlado por el jugador.
- Enemigo ligero: representa uno de los cazas enemigos.

- Enemigo pesado: representa uno de los cazas enemigos.

Conflictos

- Los enemigos tratarán de derribar al jugador.
- Al ser derribado el caza del jugador termina la partida.
- Al terminar la partida no se puede mejorar la puntuación, impidiendo mejorar la cantidad de créditos y puntos obtenidos, así como la posibilidad de posicionarse en mejores puestos del ranking de puntos.

Frontera o límite

El jugador acumula puntos hasta que su vida se agota.

Resultado

- El jugador se siente motivado a mejorar su puntuación máxima, y para ello debe seguir mejorando el caza.
- Aumento de la competitividad por alcanzar mejores lugares en el ranking de puntos.

Elementos dramáticos

Definen el entretenimiento y el nivel de inmersión de los jugadores (5). Para la solución se plantean los siguientes elementos dramáticos (24):

- Premisas: Define tiempo y lugar, relaciones entre personajes, estatus e introducción a la historia.
- Historia: Se refiere a lo que se quiere contar y dónde se va a desarrollar el juego.
- Retos: Produce una experiencia de éxito y felicidad.

Premisa

En un futuro donde la humanidad surca el cosmos en naves espaciales, una pequeña colonia es amenazada por un gran imperio. El protagonista de la historia es el jugador, que es un piloto de cazas espaciales que estaba de patrulla cuando el imperio ataca la colonia, y es su deber defenderla cueste lo que cueste y derribar a tantos cazas enemigos como sea posible.

Historia

Durante la partida no existe contenido argumental.

Reto

El jugador debe de ser capaz de mejorar su puntuación a medida que juegue más partidas y mejore su caza.

Elementos de pantallas gráficas elementales

A continuación, se muestran prototipos de las pantallas gráficas, así como la descripción de los elementos y comportamientos de estas.

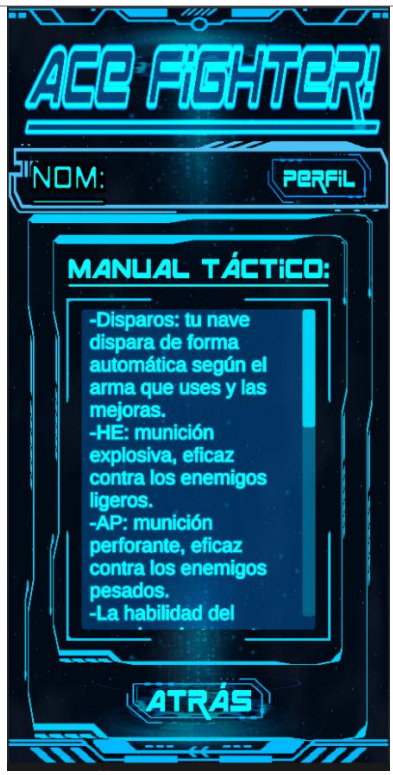
Tabla 5. Elementos de pantallas gráficas elementales.

Diseño	**Descripción**
	Interfaz inicial del juego. En ella el jugador tiene acceso a: • Perfil: despliega el panel del perfil del jugador. • Empezar: comienza el juego. • Hangar: despliega el panel del hangar o tienda. • Armería: contiene los botones necesarios para seleccionar el arma y la munición para la partida. • Ayuda: despliega el panel de ayuda. • Créditos: despliega el panel de créditos.

	Interfaz durante el juego. En ella el jugador tiene acceso a: • Botón de pausa: despliega el panel con el menú de pausa. • joystick de control: permite al jugador desplazar su caza por la pantalla. • Botón de habilidad Reparación. • Botón de habilidad Escudos. • Botón de habilidad Misiles.
	Interfaz durante juego en pausa. En ella el jugador tiene acceso a: • Reintentar: reinicia la partida. • Continuar: reanuda la partida. • Salir: regresa al menú principal.

	Interfaz mostrada cuando el jugador es derribado. En ella el jugador tiene acceso a: • Reiniciar: reinicia la partida. • Salir: regresa al menú principal.
	Interfaz mostrada cuando el jugador gestiona su perfil. En ella el jugador tiene acceso a: • Iniciar sesión: Identifica a un jugador para salvar u cargar sus datos. • Cerrar sesión: cierra la sesión y la aplicación deja de salvar los datos. • Registrar nombre: permite establecer y cambiar el nombre del jugador. • Reiniciar juego: devuelve todos los valores a su estado inicial. • Botón atrás: regresa al menú principal.

	Interfaz mostrada cuando se despliega el panel del hangar. En ella el jugador tiene acceso a: • Mejorar munición: mejora permanentemente el daño del jugador. • Mejorar cargador automático: reduce el tiempo entre disparos del jugador. • Mejorar blindaje casco: aumenta los puntos de vida del jugador. • Comprar habilidad reparación: obtiene un uso de la habilidad del mismo nombre. • Comprar habilidad escudos: obtiene un uso de la habilidad del mismo nombre. • Comprar habilidad misiles: obtiene un uso de la habilidad del mismo nombre. • Atrás: regresa al menú principal.
	Interfaz mostrada cuando se despliega el panel de créditos. En ella el jugador tiene acceso a: • En esta sección se mostrará información sobre los recursos usados en el videojuego. • Atrás: regresa al menú principal.

	Interfaz mostrada cuando se despliega el panel de ayuda. • En esta sección se mostrará información útil para el jugador sobre el contenido del juego. • Atrás: regresa al menú principal.

Especificación de mecanismos

Los mecanismos son los elementos mediante los cuales se logra el complimiento de un objetivo específico. Los mecanismos controlan distintos aspectos del juego y en conjunto definen el correcto funcionamiento de este. A continuación, se definen los mecanismos necesarios para el videojuego.

Tabla 6. Especificación de mecanismos.

No	Nombre	Descripción	Organización arquitectónica
1	Mecanismo M1 Control del menú principal	**Objetos**: Botones [Perfil, Empezar, Hangar, Créditos, Cañón, Artillería, Minigun, HE, AP, Mejorar (munición), Mejorar (cargador automático), Mejorar (Reforzar casco), Comprar (Reparación), Comprar (Escudos), Comprar (Misiles), Salir, Atrás, Reiniciar cuenta, cambiar nombre, iniciar sesión, cerrar sesión]	Mecanismo núcleo.

		Propiedades: • Todos los botones son de color azul. • Los botones al ser oprimidos parpadean una vez en azul fuerte y reproducen un sonido. • Los botones de las armas y munición al ser seleccionados se mantienen de color azul claro. **Comportamiento**: Una vez iniciado el juego, el jugador podrá interactuar con los elementos de la interfaz de usuario, los cuales le permitirán seleccionar el arma y la munición con las que quiera jugar, comenzar una partida, establecer su nombre, iniciar y cerrar sesión, reiniciar los datos del juego, acceder al hangar donde podrá adquirir mejoras y habilidades y ver los créditos. **Relaciones:** M2, M5	
2	Mecanismo M2 Control de partida	**Objetos**: Botones [Pausa, Reanudar, Reintentar, Hangar, Salir, Reparación, Escudos, Misiles], Jugador **Propiedades**: • Todos los botones son de color azul. • Los botones al ser oprimidos parpadean una vez en color blanco y reproducen un sonido. • Puntos: almacena la cantidad de puntos adquirida por el jugador. **Comportamiento**: al iniciar la partida es creado el caza del jugador con las especificaciones seleccionadas en el menú principal. El mecanismo permite al jugador pausar la partida mostrando el menú de pausa y que este pueda reanudar, reintentar o salir de la partida con las opciones contenidas en el. El mecanismo también se encarga de desplegar el menú de fin de partida cuando el jugador es derribado, permitiendo que el jugador pueda reiniciar la partida o salir. Además, el	Mecanismo núcleo.

		mecanismo es el responsable de permitir al jugador usar las habilidades que tenga disponibles, actualizar el estado de la barra de vida, el tiempo de partida. Este mecanismo es el encargado de aumentar la dificultad mediante el aumento de la vida y daño de los enemigos cada 30 segundos. **Relaciones:** M1, M5	
3	Mecanismo M3 Generación de enemigos	**Objetos**: enemigo ligero, enemigo pesado **Propiedades**: **Comportamiento**: cuando empieza la partida un temporizador se encarga de definir el momento en el que se debe crear un escuadrón de enemigos. Los escuadrones tienen diferentes formaciones, la cual es elegida de forma aleatoria. Las formaciones tienen posiciones definidas en las cuales se generará un enemigo. El tipo de enemigo que se generará en la posición es elegido aleatoriamente entre los tipos de enemigos, siempre manteniendo la simetría horizontal del escuadrón. Se debe generar un nuevo escuadrón cada 10 segundos. Si todos los enemigos son destruidos antes de 10 segundos entonces se crea un nuevo escuadrón. **Relaciones:** M4	Mecanismo núcleo.
4	Mecanismo M4 Enemigo	**Objetos**: enemigo ligero, enemigo pesado **Propiedades**: • Vida: cantidad de puntos de vida del enemigo, la cual depende del tipo de enemigo que sea. • Puntos: cantidad de puntos otorgados al jugador cuando el enemigo es destruido, los cuales dependen del tipo de enemigo que sea.	Mecanismo núcleo.

		• Velocidad: define la velocidad a la que se desplaza el enemigo por la pantalla, la cual depende del tipo de enemigo. **Comportamiento:** cuando empieza la partida y se genera un escuadrón de enemigos, este mecanismo se encarga de moverlos hasta la parte superior de la pantalla, a la cual llegarán disminuyendo su velocidad. Los enemigos mantendrán esa posición durante diez segundos y comenzarán a moverse gradualmente hacia el frente hasta salir de la pantalla. De forma aleatoria se establecerá si el enemigo realizará desplazamientos horizontales. Este mecanismo controla de los puntos de vida del enemigo y cuando estos llegan a cero destruye al enemigo y concede los puntos definidos en sus propiedades. La velocidad de los enemigos depende del tipo de enemigo. **Relaciones:** M3	
5	Mecanismo M5 Jugador	**Objetos**: Jugador **Propiedades**: • Vida: cantidad de puntos de vida que tiene el jugador. **Comportamiento**: Durante la partida gestiona el cambio de la posición del caza del jugador según la interacción del jugador con la pantalla. Limita la zona por donde el jugador puede desplazarse para evitar que salga de la pantalla. **Relaciones:**	Mecanismo núcleo.
6	Mecanismo M6 Armas	**Objetos**: Cañón, Artillería, Minigun **Propiedades**: • Proyectil: define el tipo de proyectil en dependencia del arma. • Frecuencia de disparo: tiempo que transcurrirá entre ráfagas en dependencia del arma.	Mecanismo núcleo.

		Comportamiento: Al comenzar la partida este mecanismo define el tipo de arma y los atributos correspondientes con su tipo. Un temporizador define cuando el arma es disparada y el tiempo entre los disparos de la ráfaga si es necesario. Al disparar se crea el efecto visual y sonoro de un disparo. **Relaciones:** M7	
7	Mecanismo M7 Proyectil	**Objetos**: proyectil **Propiedades**: • Daño: define la cantidad de puntos de vida que se le restará al objetivo impactado por el proyectil. **Comportamiento:** este mecanismo define qué tipo de proyectil es disparado en dependencia del arma que lo disparó y sus estadísticas. El mecanismo también añade pequeñas desviaciones a la trayectoria de cada proyectil. Además, se encarga de crear el efecto visual y sonoro de una explosión al momento de impacto entre los proyectiles con los enemigos o el jugador. **Relaciones:** M6, M8	Mecanismo núcleo.
8	Mecanismo M8 Recompensa	**Objetos**: enemigo ligero, enemigo pesado **Propiedades**: **Comportamiento:** este mecanismo se encarga de llevar el control de la puntuación del jugador y recompensarlo en dependencia con cierta cantidad de créditos. **Relaciones:** M9	Mecanismo núcleo.
9	Mecanismo M9 Movimiento del escenario	**Objetos**: Fondo **Propiedades**: • Velocidad: define la velocidad a la que se desplazará el fondo para dar la sensación de movimiento continuo perpetuo al resto de los objetos de la escena.	Mecanismo núcleo.

		Comportamiento: al iniciar la partida crea el fondo del escenario y se encarga de modificar sus propiedades para moverlo indefinidamente hacia abajo de la pantalla. Una vez deja de ser visible el objeto de fondo, el mecanismo lo elimina, no sin antes crear otro para dar la sensación de que se trata de un fondo infinito. **Relaciones:**	
10	Mecanismo M10 Conexión	**Objetos**: **Propiedades**: **Comportamiento:** este mecanismo se encarga de establecer la comunicación e intercambio de datos con la plataforma Cosmox. Debe comprobar que los datos del usuario introducido existan en la plataforma y enviar los puntos del jugador para mantener actualizado el ranking. **Relaciones:**	Mecanismo alternativo.

Requisitos no funcionales

Los requerimientos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando [41].

- RNF 1 Usabilidad
  - Cualquier usuario debe ser capaz de jugar el juego.
  - Debe proteger al usuario de cometer errores.
- RNF 2 Seguridad
  - Debe garantizar la protección de los datos ingresados por el usuario para consumir los servicios de Cosmox.
- RNF 3 Compatibilidad
  - El videojuego debe coexistir con otras aplicaciones ejecutadas en el mismo sistema y compartir los recursos.
- RNF 4 Fiabilidad
  - El videojuego debe estar disponible en todo momento y funcionar correctamente cuando el usuario haga uso de este.

- RNF 5 Portabilidad
    - Debe ser capaz de ejecutarse correctamente en la versión de Android 5.0 o superior.

Paquetes de mecanismos

Un paquete es un mecanismo que permite organizar los elementos modelados con UML, facilitando de esta forma el manejo de los modelos de un sistema complejo [42]. Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. Los paquetes pueden estar anidados unos dentro de otros, y unos paquetes pueden depender de otros paquetes. Se pueden utilizar para plantear la arquitectura del sistema a nivel macro. Muestra cómo está estructurado el sistema, cada paquete puede contener otros paquetes o clases, que tienen interfaces y realizan cierta funcionalidad. También se pueden mostrar algunas clases dentro de los paquetes, así como las relaciones de dependencia de estas clases con otras clases o paquetes.

- Mecanismos Núcleo: representan a los mecanismos centrales del videojuego que constituyen el medio para captar la atención jugador y el propósito esencial del juego. Permiten la interacción del jugador con cada uno de los elementos que contienen las escenas, así como la navegación por ellas, con el propósito de cumplir con los niveles del juego [43].
- Mecanismo Alternativo: representan a los mecanismos que aportan al propósito general del juego, pero no constituyen elementos fundamentales de este para su funcionamiento. Permiten la incorporación de elementos que enriquecen las opciones de interacción con el videojuego.

A continuación, se muestra el diagrama de paquetes de mecanismos donde se representa el propósito general del juego:

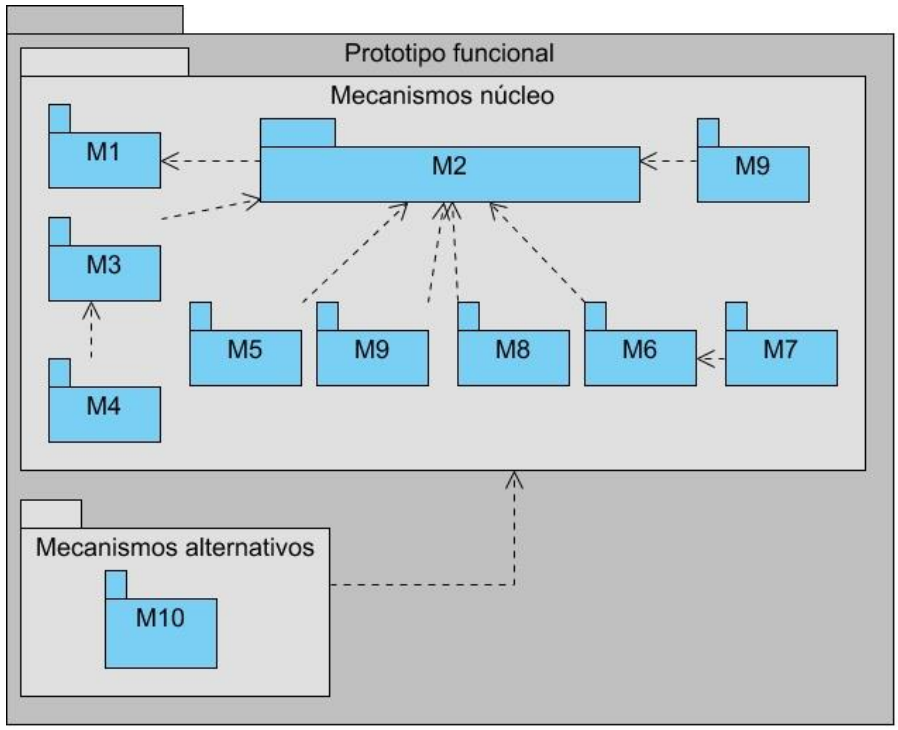


Figura 5. Diagrama de paquetes de mecanismos.

- M1 Control del menú principal.
- M2 Control de partida.
- M3 Generación de enemigos.
- M4 Enemigo.
- M5 Jugador.
- M6 Armas.
- M7 Proyectil.
- M8 Recompensa.
- M9 Movimiento del escenario.
- M10 Conexión.

Estilo arquitectónico

Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles o rasgos de los elementos arquitectónicos y las relaciones permitidas entre ellos dentro de la arquitectura

que se conforma a ese estilo [44]. El estilo llamada y retorno se caracteriza por la descomposición jerárquica en subrutinas (componentes) que solucionan una tarea o función definida. Los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. Este estilo arquitectónico permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala y se basa en la bien conocida abstracción de procedimientos, funciones y métodos.

Arquitectura de software

La arquitectura de la solución propuesta incluye los elementos fundamentales de la tesis “Arquitectura de software para videojuegos desarrollados sobre el motor de juego Unity3D” [45]. La arquitectura está compuesta por la combinación de los patrones arquitectónicos: arquitectura basada en capas y arquitectura basada en componentes, para estructurar los elementos del videojuego. Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces definidas o instancias de clases [45]. Para cada componente se definió una clase con sus atributos y operaciones para ejecutar los métodos y acciones relevantes. Para que las clases puedan comunicarse con otras en dependencia de la capa donde se encuentren fueron definidas las interfaces e instancias necesarias.

Distribución de las capas presentes en la arquitectura:

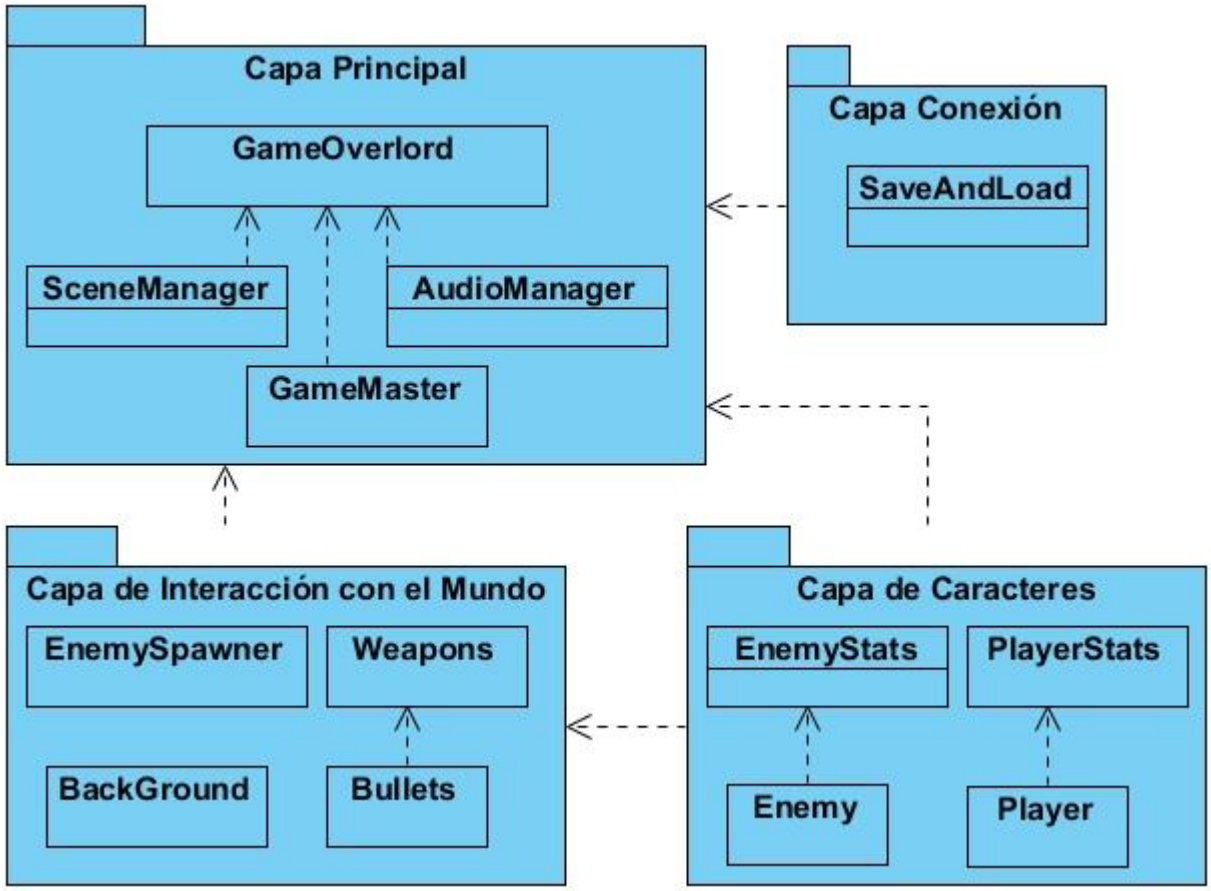


Figura 6. Capas de la arquitectura de la propuesta de solución.

Capa Principal: en esta capa se encuentran los controladores principales del juego.

- GameOverlord: es el controlador encargado del funcionamiento del juego. Guarda instancias de los elementos necesarios en todas las escenas y las mantiene disponibles para su uso. Se encarga de las transiciones entre escenas y el tratamiento de estos.
- GameMaster: es el controlador encargado del funcionamiento de la partida, la dificultad y los elementos de la interfaz de esta. Se implementa como un GameObject en la escena de juego. Guarda instancias de los enemigos y los elementos de la interfaz.
- SceneManager: es el controlador encargado de gestionar las escenas. Tiene las funcionalidades para cambiar o recargar las escenas.

- AudioManager: es el controlador encargado de reproducir la música y los efectos de sonido cuando sean necesarios.

Capa de Interacción con el Mundo: en esta capa se encuentran los componentes que permiten el desarrollo del videojuego mediante la interacción del jugador.

- EnemySpawner: es el controlador encargado de instanciar las formaciones enemigas generadas proceduralmente.
- Weapons: es el controlador que se encarga de definir las estadísticas de cada tipo de arma instanciada.
- Bullets: es el controlador que se encarga de definir las estadísticas de cada tipo de proyectil instanciado.
- BackGround: es el controlador encargado de instanciar, mover y destruir el fondo del juego.

Capa de Caracteres: en esta capa se encuentran los componentes que permiten el control de los elementos en el videojuego.

- PlayerStats: es el controlador encargado de almacenar las estadísticas del jugador.
- Player: es el controlador que se encarga de modificar la posición del jugador y controlar de los puntos de vida.
- EnemyStats: es el controlador que define las estadísticas de los enemigos.
- Enemy: es el controlador que se encarga definir el patrón de movimiento de los enemigos y controlar de los puntos de vida de estos.

Capa de Conexión: en esta capa se encuentran los componentes que permiten la conexión con la plataforma Cosmox.

- SaveAndLoad: es el controlador que establece la conexión con la plataforma Cosmox e intercambia datos con esta, permitiendo el uso de los servicios que brinda la plataforma.

Diagrama de clases

Para una mejor comprensión de la forma en que se estructura el sistema de clases de la propuesta de solución, se muestra el siguiente diagrama de clases perteneciente al mecanismo M1 Control de Interfaz de Usuario en Juego. Los diagramas de clases del resto de mecanismos se encuentran en el anexo A1.

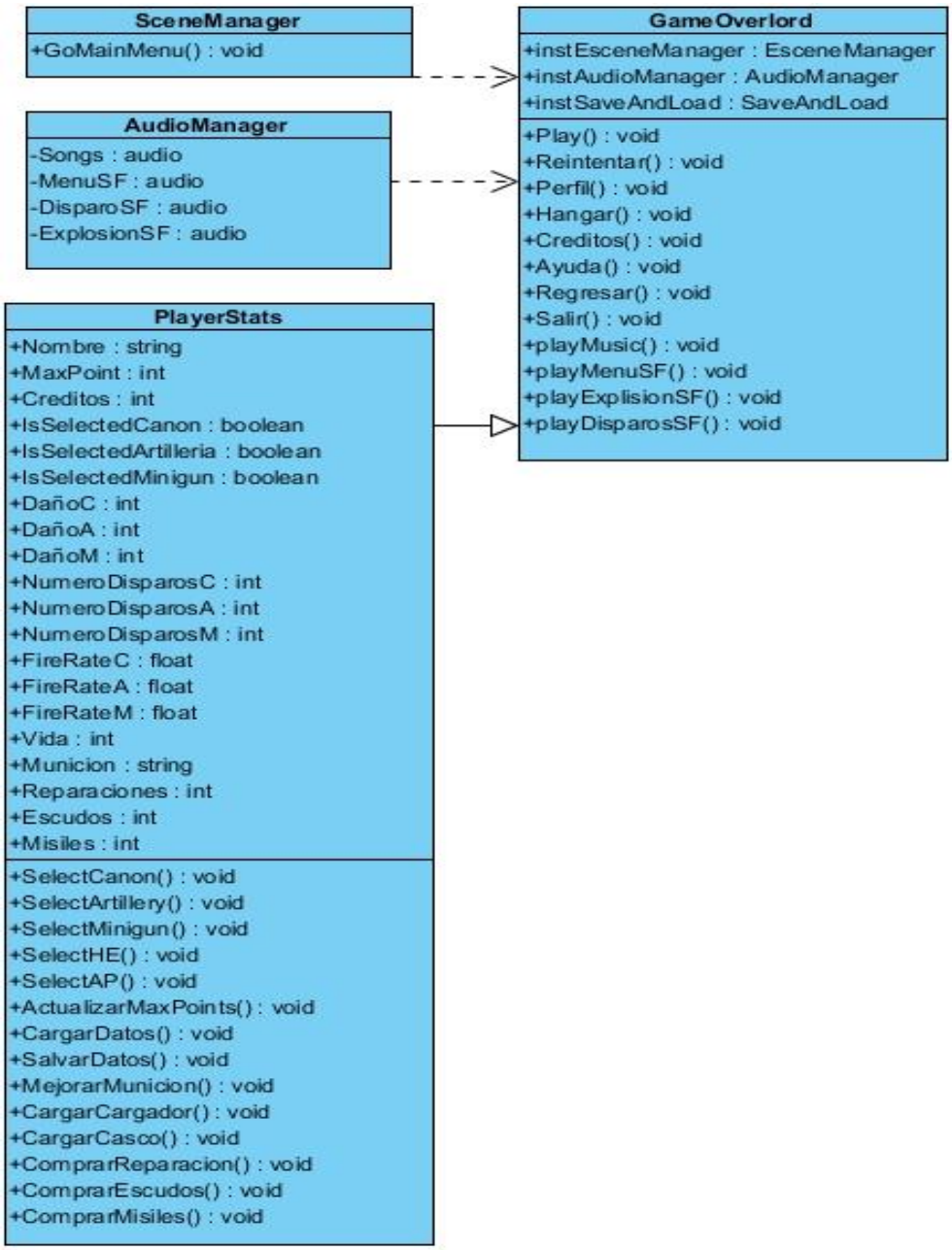


Figura 7. Diagrama de clases del mecanismo M1 Control de menú principal.

Patrones de diseño

Un patrón de diseño proporciona un esquema para refinar los subsistemas o componentes de un sistema software, o las relaciones entre ellos [46]. Describe estructuras repetitivas de comunicar componentes que

resuelven un problema de diseño en un contexto particular. Es una información nombrada, instructiva e intuitiva que captura la estructura esencial y la perspicacia de una familia de soluciones probadas con éxito para un problema repetitivo que surge en un cierto contexto y sistema. En el diseño de la solución se utilizarán los Patrones Generales de Software de Asignación de Responsabilidad (GRAPS) y la Banda de Cuatro (GOF).

Patrones GRASP usados en la solución:

- Experto en información: Suele conducir a diseños donde los objetos se hacen responsables de las mismas operaciones. Tiene como beneficio que se mantiene el encapsulamiento de la información, lo que posibilita menor acoplamiento. Se distribuye el comportamiento entre las clases que contienen la información requerida, posibilitando clases más ligeras logrando mayor cohesión. El uso de la herramienta Unity facilita el uso de este patrón ya que se puede asignar como un elemento de los GameObjects el fragmento de código que establecerá su comportamiento o tareas como en el caso de los proyectiles o el EnemySpawner.
- Alta cohesión: Es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo superior al estrictamente necesario. Una clase con baja cohesión tiene muchas responsabilidades con poca relación entre sí, haciéndola difícil de comprender, reutilizar y conservar, mientras que una alta cohesión favorece la claridad del sistema y disminuye el acoplamiento [47]. En la implementación de la solución esto se evidencia en la separación de las responsabilidades de la clase GameMaster que dieron lugar a la implementación de las clases EnemySpawner y BackGround para especializar sus responsabilidades.
- Bajo acoplamiento: Es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Una clase con baja cohesión tiene muchas responsabilidades, difíciles de entender, reutilizar, mantener, delicadas y frágiles. Tiene como beneficio que incrementa la claridad y comprensión del sistema [47], se simplifica el mantenimiento, favorece una alta cohesión y facilita la reutilización. Por ejemplo, las armas de los enemigos y el jugador no son controladas por estos elementos, esa responsabilidad se divide en las armas y los proyectiles, guardando relación con el uso del patrón Experto en información.

- Controlador: Este patrón se manifiesta mediante un objeto responsable del manejo de los eventos del sistema, que no pertenece a la interfaz de usuario. El controlador recibe la solicitud del servicio desde la interfaz y coordina su realización delegando a otros objetos. En el proceso de desarrollo se utiliza la clase GameOverlord, que se encarga de las acciones en el software.
- Creador: Este patrón es el responsable de asignarle a una clase determinada la responsabilidad de crear una instancia de las demás clases referentes al sistema. El mismo se evidencia en la clase EnemySpawner y Weapons, donde las mismas son responsables de crear las instancias de clases necesarias que están implicadas en la creación de los enemigos y los disparos.

Patrones GOF usados en la solución:

- Única instancia (Singleton): El patrón de diseño Singleton se asegura de que una clase solo tenga una instancia y provee acceso global a ella. El uso de este patrón se evidencia en las clases PlayerStats, GameOverlord, SaveAndLoad, SceneManager y SoundManager, ya que son las clases con información de los elementos del videojuego que se necesitan estén accesibles en todo momento, permitiendo la optimización de código y la sobrecarga de los recursos.

  En la solución se implementó el patrón Singleton en la clase GameOverlord, ya que esta es la encargada de brindar un medio de comunicación entre distintas clases y de mantener una instancia única de sí misma para que no haya datos duplicados. El empleo de este patrón permite la persistencia de los datos entre escenas y los mantiene públicos para su uso por otras clases del sistema.

```
//*----------------Singleton-----------------------*//
2 references
public static GameOverlord instance;
0 references
private void Awake()
{

    if (GameOverlord.instance == null)
    {
        GameOverlord.instance = this;
    }
    else
    {
        Destroy(gameObject);
    }
    DontDestroyOnLoad(gameObject);
}
//*----------------Singleton-----------------------*//
```

Figura 8. Patrón de diseño Singleton implementado en la solución.

- Fachada (Facade): El patrón Fachada provee una interfaz a una serie de clases para evitar el uso de varias instancias, dejando a las subclases accesibles para ser usadas directamente. Se implementa en el GameOverlord, que junto al patrón Singleton es la encargada de gestionar los elementos principales del videojuego, permitiendo el acceso al resto de las clases asociadas a ella.

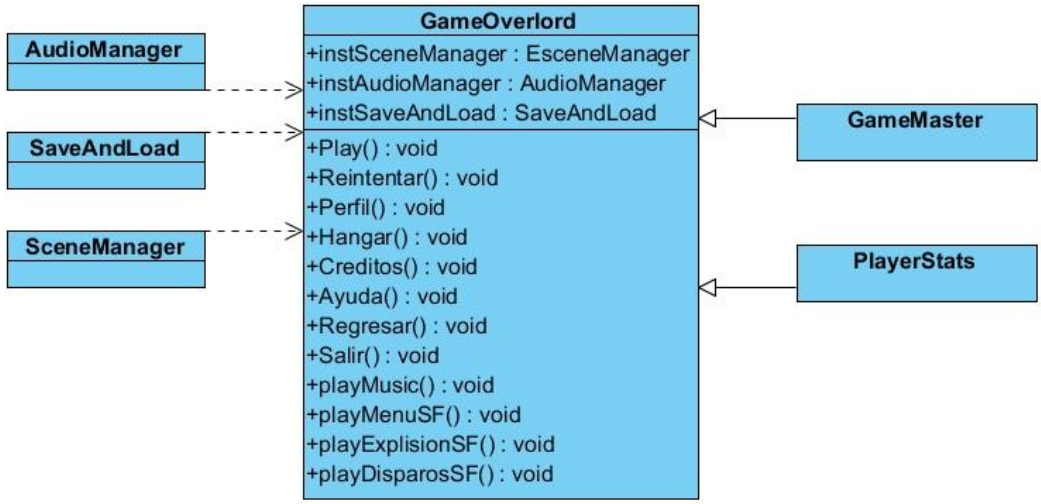


Figura 9. Patrón de diseño Fachada implementado en la solución.

Representación de comportamiento

Para representar el comportamiento del videojuego según la acción del jugador se emplean los diagramas de estados. En la Figura 10 se muestra el diagrama de estados del mecanismo M2 Control de partida, donde se muestran las posibles acciones que puede realizar el jugador y los estados destino a los que puede llegar en este escenario. El resto de los diagramas de estados se encuentran en el anexo A2.

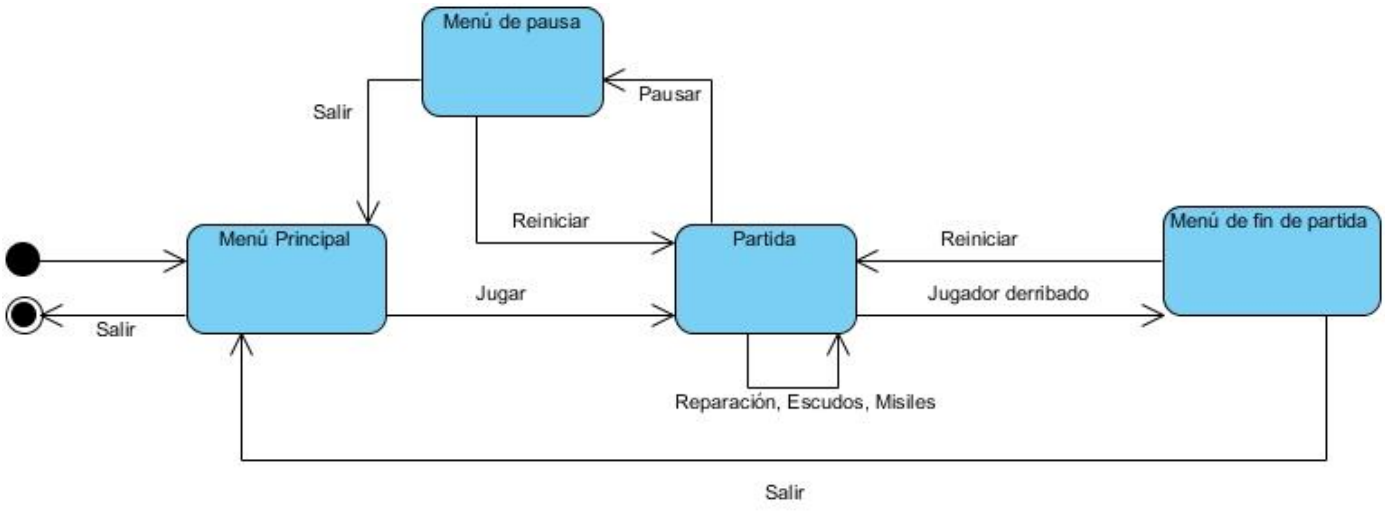


Figura 10. Diagrama de estado del mecanismo M2 Control de partida.

Conclusiones parciales del capítulo

Al terminar el proceso de diseño se obtuvo una representación lógica de los procesos, mecanismos y estructura arquitectónica para la creación de un prototipo funcional de la propuesta de solución. Adicionalmente se describieron los principales requisitos no funcionales y elementos formales que permiten representar las características físicas y una base lógica para el proceso de desarrollo de software.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DE LA SOLUCIÓN

En el presente capítulo se representa la estructura del videojuego a nivel de componentes. Se describe el estándar de codificación, el algoritmo de los componentes más relevantes y se explican los resultados obtenidos en las pruebas alfa y beta.

3.1 Estándar de codificación

En la implementación de la solución se usó el estilo de escritura CamelCase. El nombre viene porque se asemeja a las dos jorobas de un camello, y se puede dividir en dos tipos [34]:

- UpperCamelCase: usado en la declaración de métodos y clases de la solución.
- LowerCamelCase: usada en la declaración de variables.

Vista del código

Definición de clases: Las clases comienzan con inicial mayúscula al inicio de la palabra y en caso de estar conformada por palabras compuestas, la definición debe ser continua y cada palabra debe iniciar con mayúscula siguiendo el estándar determinado. Un ejemplo de ello se muestra a continuación:

```
public class GameMaster : MonoBehaviour
```

Declaración de métodos: El nombre de los métodos debe comenzar con mayúscula y en caso de que el nombre esté compuesto por más palabras cada una debe iniciar con mayúscula. A continuación, se presenta un ejemplo de cada caso:

```
void SetActivWeapon()
```

```
public void TakeDamege(int damage, GameObject proyectil){
```

Declaración de variables: Los nombres de las variables comienzan con minúscula y en caso de estar compuesto por más palabras la segunda debe empezar con mayúscula siguiendo el estándar determinado, como se muestra en los siguientes ejemplos:

```
public int vida;
3 references
public float timepoDeEscudo = 10f;
4 references
public GameObject escudosDeEnergia;
```

Estilos de indentación: El estilo utilizado en la implementación de la propuesta de solución es propio para lenguajes de programación que usan llaves para delimitar bloques lógicos de códigos. A continuación se presenta un ejemplo de su utilización:

```
for (int i = 0; i < objetivos.Length; i++)
{
    if (Vector3.Distance(transform.position, objetivos[i].transform.position) < rangoImpacto)
    {
        TargetToDamage = objetivos[i];
    }
}
```

3.2 Implementación de los algoritmos relevantes.

A continuación, se describirá la forma en que fueron implementadas las funciones más relevantes del videojuego. Durante el desarrollo del epígrafe se hará referencia a aspectos específicos relacionado al empleo del motor de videojuego Unity. Para un mejor entendimiento se recomienda consultar la documentación oficial de Unity [48].

Algoritmo para el incremento de dificultad

La dificultad del videojuego se incrementa cada 30 segundos mediante el aumento de las estadísticas de daño y vida de los enemigos.

1. Crear una variable X para almacenar la cantidad de tiempo que debe de transcurrir para aumentar la dificultad.

```
public float tiempoParaAumentarDificultad = 30f;
```

2. Declarar el método Y encargado de acceder a la clase donde se guardan las estadísticas con las que son instanciados los enemigos y aumentar sus valores según la cantidad especificada en la especificación de mecanismos.

```
public void AumenarDificultad(){
    EnemyWeaponStats.DamageArtillery += 10;
    EnemyWeaponStats.DamageCanon += 10;
    EnemyWeaponStats.DamageMinigun += 10;
    EnemyWeaponStats.vidaLigero += 50;
    EnemyWeaponStats.vidaPesado += 50;
}
```

3. Disminuir el valor de la variable X el tiempo con el tiempo transcurrido.

```
tiempoParaAumentarDificultad -= Time.deltaTime;
```

4. Cuando el valor de la variable X sea menor que cero llamar al método Y.
5. Tras la ejecución del método Y restaurar el valor de la variable X a su valor original.

```
if (tiempoParaAumentarDificultad <= 0)
{
    AumenarDificultad();
    tiempoParaAumentarDificultad = 30f;
}
```

Generación procedural de las formaciones enemigas

Esta función perteneciente al mecanismo M3 Generación de enemigos es implementada en la clase *EnemySpawner*. La función de esta clase es la de crear las formaciones de enemigos cada cierto tiempo definido por el desarrollador o cuando todos los enemigos de la pantalla sean eliminados.

1. Declarar las variables necesarias.
    - *cantidadDeEnemigosEnPantalla*: para tener el control de los enemigos presentes en pantalla.
    - *enemigoNormal* y *enemigoPesado*: estas variables son del tipo *GameObject* presente en Unity para almacenar componentes y utilizarlos posteriormente.
    - *tiempoEntreOlas*: se usa para controlar el tiempo que debe transcurrir antes de crear una nueva formación.
    - *valorResetearTiempoEntreOlas*: se utilizó para guardar el valor al cual se debe reiniciar el valor de la variable *tiempoEntreOlas* tras la creación de una formación.

- *enemySpawnPositionNormal*: esta variable se utiliza para guardar el valor fijo de la posición a partir de la cual se generarían las formaciones. La variable es del tipo *Transform* y se emplea dentro de Unity para guardar el valor de posición en el espacio representado por un vector con coordenadas en los ejes x, y, z.

```
public int cantidadDeEnemigosEnPantalla = 0;
2 references
public GameObject enemigoNormal;
2 references
public GameObject enemigoPesado;
5 references
public Transform enemySpawnPositionNormal;
4 references
public float tiempoEntreOlas = 0f;
1 reference
public float valoResetearTiempo = 5f;
```

2. Declarar el método encargado de seleccionar la formación de forma aleatoria.
    - *SpawnearEscuadron(int formacion)*: el método recibe como parámetro un número aleatorio determinado por la función *Random.Range(1,3)* de Unity, la cual devuelve un valor aleatorio entre los parámetros que recibe. El número aleatorio es guardado en la variable entera *formacion*.
    - Se declararon dos variables locales del tipo *GameObject* para almacenar una referencia al componente de tipo *enemigoNormal* o *enemigoPesado*, con el objetivo de seleccionar aleatoriamente al tipo de enemigo que será creado en el centro de la formación y los que serán creados en los extremos, si esta es de una cantidad de elementos impar. Para la selección del enemigo se empleó otro método.
    - Se crea una estructura *switch* en la que se evalúa el valor de la variable *formacion*. Cada caso del *switch* corresponde al número de una formación y dentro del bloque de código correspondiente se especifica la posición y enemigo a instanciar. Además de aumentar el valor de la variable *cantidadDeEnemigosEnPantalla* por la cantidad de enemigos instanciados.

```
void SpawnearEscuadron(int formacion)
{
    GameObject WingLider = EnemigoASpawear();
    GameObject Escolta1 = EnemigoASpawear();

    switch (formacion)
    {
        case 1:
            Instantiate(WingLider, new Vector3(0, 0, 250), enemySpawnPositionNormal.rotation);
            Instantiate(Escolta1, new Vector3(35, 0, 250), enemySpawnPositionNormal.rotation);
            Instantiate(Escolta1, new Vector3(-35, 0, 250), enemySpawnPositionNormal.rotation);
            cantidadDeEnemigosEnPantalla += 3;
            break;
        case 2:
            Instantiate(Escolta1, new Vector3(20, 0, 250), enemySpawnPositionNormal.rotation);
            Instantiate(Escolta1, new Vector3(-20, 0, 250), enemySpawnPositionNormal.rotation);
            cantidadDeEnemigosEnPantalla += 2;
            break;
        default:
            Debug.Log("404");
            break;
    }
}
```

3. Declaración del método encargado de la selección aleatoria de un enemigo entre los disponibles.
    - *EnemigoASpawnear()*: al contar con una lista de solo dos enemigos disponibles, este método genera dentro de una condicional un numero aleatorio entre 1 y 3 y lo compara con 1, de ser verdadera la condición el enemigo seleccionado es el *enemigoNormal* y si es falsa se selecciona al *enemigoPesado*.

```
GameObject EnemigoASpawear()
{
    if (Random.Range(1, 3) == 1)
    {
        return enemigoNormal;
    }
    else
    {
        return enemigoPesado;
    }
}
```

4. Disminuir en el tiempo la variable *tiempoEntreOlas*.

```
tiempoEntreOlas -= Time.deltaTime;
```

5. Si la variable *tiempoEntreOlas* es menor que cero, llamar al método *SpawnearEscuadron()*

```
if (tiempoEntreOlas <= 0)
{
    SpawnearEscuadron(Random.Range(1, 3));
    tiempoEntreOlas = valoResetearTiempo;
}
```

6. Si el valor de la variable *cantidadDeEnemigosEnPantalla* es igual o menor a 0, establecer el valor de la variable *tiempoEntreOlas* igual a 0, esto permitirá que se cumpla la condición anterior.

```
if (cantidadDeEnemigosEnPantalla <= 0)
{
    tiempoEntreOlas = 0f;
}
```

7. Restaurar el valor de la variable *tiempoEntreOlas* al valor guardado en la variable *valorResetearTiempoEntreOlas* una vez creada la formación.

```
tiempoEntreOlas = valoResetearTiempo;
```

Método para consumir el servicio de ranking de Cosmox

Esta función pertenece al mecanismo M10 Conexión y es empleada en dos momentos, primero para la identificación del usuario y segundo para enviar la puntuación obtenida por el usuario en la partida a Cosmox. Se omitirá la inclusión del código para evitar la distribución de información sensible. A continuación, se describe el algoritmo empleado:

1. Se declararon las variables necesarias dentro de las que se incluye:
    - Una variable para controlar si el usuario ha sido identificado o no en Cosmox.
    - Variables para almacenar el usuario y el pin del usuario si este ha sido identificado.
    - Variables para guardar la identificación del videojuego en la plataforma y la dirección url a la que debe acceder para usar el servicio.
    - Variables del tipo entrada de texto para almacenar recoger la información entrada por el usuario al identificarse.
2. Se implementaron los métodos para iniciar conexión y enviar la puntuación. La diferencia entre ambos métodos radica en el momento que son llamados y los datos que envían a Cosmox. El

método para empezar la conexión solo envía los datos de nombre de usuario y el pin del usuario, si la conexión es exitosa se guardan estos datos para su posterior uso en el método para enviar la puntuación.

- Ambos métodos comienzan con la estructuración de los datos que se van a enviar en una cadena de caracteres siguiendo una estructura predefinida.
- La estructura es codificada empleando las funciones de una clase proporcionada por Vertex que hace uso de una llave específica.
- Se hace uso de las funcionalidades de Unity para mandar datos a través de la web y la información de las variables, se envía a la url predefinida la estructura de datos codificada como una cadena de texto.
- El paso anterior devuelve un conjunto de datos como respuesta, los cuales son almacenados para su tratamiento y de los cuales se puede obtener un conjunto de datos que permite determinar si la acción se realizó con éxito o si existió algún error en el proceso.

3.3 Assets y recursos utilizados

Assets

Como parte de la implementación del mecanismo M2 Control de partida, se utiliza un elemento de interfaz para controlar el movimiento del caza del jugador bajo una licencia estándar de usuario final (EULA). El elemento conocido como joystick, forma parte del paquete Joystick Pack, disponible en la Unity Assets Store, el cual está pensado para incorporar de forma rápida y sencilla controles para videojuegos basados en el sistema Android [49].

Efectos de sonido

Los efectos de sonidos fueron obtenidos de Kenney.nl, un portal web donde se pueden encontrar cientos de assets, sprites, sonidos y modelos 3D bajo licencia de CC0 Royalty Free, la cual permite su uso en todo tipo de proyectos, incluso comerciales [50].

Imágenes

Las imágenes utilizadas para el diseño de la interfaz de usuario fueron obtenidas en Vecteezy.com bajo la licencia Creative Commons attribution [51]. Este sitio ofrece una gran variedad de imágenes vectoriales de

todo tipo. Las imágenes seleccionadas fueron modificadas en la herramienta GIMP para añadir transparencia y cambiarle el color a blanco, con el fin de aumentar su versatilidad dentro del motor de juego Unity.

3.4 Diagramas de componentes

En la Figura 11 se presenta el diagrama de componentes perteneciente al mecanismo M1 Control del menú principal. Con el mismo se puede apreciar desde una vista más cercana al código, cómo el videojuego es estructurado en diferentes componentes que se relacionan entre sí. La representación de los demás mecanismos se encuentra en el anexo A3.

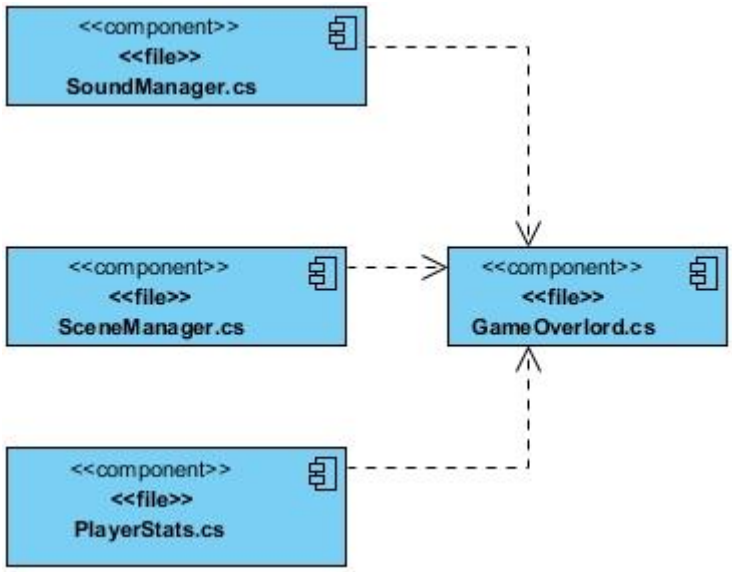


Figura 11. Diagrama de componentes M1 Control del menú principal.

3.5 Pruebas

Al concluir el proceso de desarrollo del videojuego y encontrarse en estado funcional, se desarrolló un conjunto de pruebas validar todos los mecanismos. Las pruebas aplicadas se hicieron tomando en cuenta los pasos descritos por la metodología seleccionada, que trabaja sobre la base del análisis del prototipo compilado.

Pruebas alfa y beta

- Pruebas Alpha: estas pruebas las realiza el desarrollador. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y problemas de uso. Las pruebas alfa se hacen en un entorno controlado [35].
- Pruebas Beta: estas pruebas las realizan los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente [35].

Pruebas Alpha

A continuación, se describe el registro de defectos encontrados durante las pruebas Alpha:

- Iteración 1: se realizan pruebas a los mecanismos M1 Control del menú principal y M10 Conexión.
    - NC 1.1: No existe retroalimentación del arma o munición seleccionada.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: panel de selección de armas y munición.
        - Tipo de error: app-error de interfaz.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.2: Error ortográfico.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: panel de opciones del perfil, la palabra “sesión” no están acentuadas.
        - Tipo de error: app-Doc/ortografía.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.3: No se muestra el precio correcto de los objetos del hangar.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: panel del hangar.
        - Tipo de error: app-error de interfaz.
        - Impacto: alto.
        - Estado del defecto: resuelto.

        - Respuesta del equipo: resuelto.
    - NC 1.4: Reiniciar los datos no reinicia el nombre y los puntos mostrados.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: panel del menú principal.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
- Iteración 2: se realizan pruebas a los mecanismos M2 Control de partida, M3 Generación de enemigos, M4 Enemigo, M5 Jugador y M9 Movimiento del escenario.
    - NC 2.1: El jugador puede exceder los límites de la pantalla.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 2.2: Solo se genera un tipo de enemigo.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 2.3: El botón Hangar no funciona como es debido.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: panel del menú de fin de partida.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.

        - Respuesta del equipo: resuelto.
    - NC 2.4: Los botones de las habilidades siguen funcionales aun cuando el juego está en pausa o se terminó la partida.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
- Iteración 3: se realizan pruebas a los mecanismos M6 Armas, M7 Proyectil y M8 Recompensa.
    - NC 3.1: Todas las armas se comportan igual.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 3.2: No existe retroalimentación del impacto de los proyectiles.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-error de interfaz.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 3.3: El juego no termina cuando el jugador es derribado.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.

        - Respuesta del equipo: resuelto.
    - NC 3.4: Los enemigos derribados no suman puntos al total de puntos de la partida.
        - Nombre del probador: Oscar Mastrapa Prats.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.

Pruebas Beta

Para la confección de las pruebas Beta, se selecciona un equipo de jugadores para que prueben el funcionamiento del videojuego, los cuales se relacionan a continuación:

- Lázaro Ernesto Martínez Gonzales (estudiante de 5to año en Ingeniería en Ciencias Informáticas, facultad 1).
- Pedro José Ramírez González (estudiante de medicina).
- Álvaro Sierra Ariad (estudiante de arquitectura).
- Mirel Alberto Feria Martínez (Ingeniero Industrial).
- Ernesto Alfredo Estrada Casas (estudiante de 5to año en Ingeniería en Ciencias Informáticas, facultad 1).

Se realizaron tres iteraciones de las pruebas Beta, cuyos resultados pueden ser consultados en la sección de los anexos A4. Las no conformidades arrojadas por las pruebas en una iteración fueron resueltas antes de iniciar la siguiente. En la tercera iteración se registran cero no conformidades por parte del personal que realizó las pruebas, por lo que no fue necesario realizar otra iteración. En la Figura 12 se muestra un resumen de este proceso de validación.

Figura 12 No conformidades.

Conclusiones parciales

Tras realizar el proceso de implementación y pruebas se arribaron a las siguientes conclusiones:

- La descripción precisa de los mecanismos y los diagramas de componentes permitieron la comprensión e implementación del código de forma más sencilla.
- Se logró la implementación de un método para la generación de formaciones enemigas.
- Se logró la implementación de un método para el aumento progresivo de dificultad.
- Se implementó un mecanismo capaz de hacer uso del servicio de ranking de Cosmox.
- La descripción correcta de las pruebas alfa y beta permitió la detección de elementos que afectaban a la calidad y desempeño del videojuego.

CONCLUSIONES:

Tras culminar la investigación y el proceso de desarrollo de la propuesta de solución se arribaron a las siguientes conclusiones:

- Se obtuvo una versión estable y funcional del videojuego Ace Fighter, el cual es un producto para dispositivos con sistema Android y enriquece el catálogo de videojuegos casuales de la industria nacional.
- Se implementó un método para consumir el servicio de ranking de Cosmox, permitiendo la competitividad entre jugadores.
- Se desarrolló satisfactoriamente un método de generación procedural de contenido para crear aleatoriamente las formaciones de enemigos durante el juego.
- Se desarrolló un método para el aumento progresivo de dificultad.

RECOMENDACIONES:

A partir de esta primera entrega y como indica la metodología empleada, se puede retomar la parte de diseño y conceptualización para mejorar y añadir contenido al videojuego.

En vista al futuro desarrollo del videojuego se recomienda:

- Creación de un sistema de niveles, donde cada uno conste de un tiempo y dificultad predeterminada.
- Incorporación de más contenido en formato de naves para el jugador, enemigos, armas y habilidades.

REFERENCIAS BIBLIOGRÁFICAS:

[1] *The History Of Gaming Consoles - An Industry Decades In The Making - Tell Me Best* [online]. [vid. 2022-06-19]. https://www.tellmebest.com/history-of-gaming-consoles/

[2] *Géneros de videojuegos* | *Wikijuegos* | *Fandom* [online]. [vid. 2022-06-19]. https://videojuegos.fandom.com/es/wiki/G%C3%A9neros_de_videojuegos

[3] *COSMOX* | *Videojuego Coliseum* [online]. [vid. 2022-07-04]. https://cosmox.uci.cu/frontend/web/videogame/40

[4] *COSMOX* | *Ranking del videojuego La Neurona 2* [online]. [vid. 2022-07-04]. https://cosmox.uci.cu/frontend/web/videogame/view-ranking?id=34

[5] *COSMOX* | *Ranking del videojuego Kuba Kart* [online]. [vid. 2022-07-04]. https://cosmox.uci.cu/frontend/web/videogame/view-ranking?id=39

[6] *Candy Crush Saga - Aplicaciones en Google Play* [online]. [vid. 2022-07-04]. https://play.google.com/store/apps/details?id=com.king.candycrushsaga&hl=es&gl=US

[7] *Subway Surfers - Aplicaciones en Google Play* [online]. [vid. 2022-07-04] https://play.google.com/store/apps/details?id=com.kiloo.subwaysurf&hl=es&gl=US

[8] *COSMOX* | *Videojuego La Súper Claria* [online]. [vid. 2022-07-04]. https://cosmox.uci.cu/frontend/web/videogame/25

[9] *COSMOX* | *Videojuego La Chivichana* [online]. [vid. 2022-07-04]. https://cosmox.uci.cu/frontend/web/videogame/29

[10] HERNÁNDEZ P., Andy; Pérez T. Karina; Correa M., Omar. Marco de trabajo ingenieril para el proceso de desarrollo de videojuegos (Engineering framework for the videogame development process). *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software RACCIS* [online]. 2017, **7**(1), 13–26. doi:10.5281/zenodo.2617305

[11] MERIEUX, Antonin. *Le Jeu et l'Homme : une approche épistémologique* [online]. B.m., 2016 [vid. 2022-06-19]. These de doctorat. Paris 4. https://www.theses.fr/2016PA040117

[12] *FrascaThesisVideogames.pdf* [online]. [vid. 2022-06-19]. https://ludology.typepad.com/weblog/articles/thesis/FrascaThesisVideogames.pdf

[13] *(PDF) Breve historia de los videojuegos* [online]. [vid. 2022-06-19]. https://www.researchgate.net/publication/39499381_Breve_historia_de_los_videojuegos

[14] STAFF, GameSpot. GDC '08: Are casual games the future? *CNET* [online]. [vid. 2022-06-19]. https://www.cnet.com/tech/gaming/gdc-08-are-casual-games-the-future/

[15] PUERTA-CORTÉS, Diana Ximena, Tayana PANOVA, Xavier CARBONELL a Andrés CHAMARRO. How passion and impulsivity influence a player's choice of videogame, intensity of playing and time spent playing. *Computers in Human Behavior* [online]. 2017, **66**, 122–128. ISSN 0747-5632. doi:10.1016/j.chb.2016.09.029

[16] *Shoot'em Up | Definición en GamerDic* [online]. [vid. 2022-06-19]. https://www.gamerdic.es/termino/shootem-up

[17] ADAMS, Ernest a Andrew ROLLINGS. *Fundamentals of game design*. Pearson Prentice Hall: Upper Saddle River, NJ, 2006. ISBN 978-0-13-168747-9.

[18] PARKIN, Simon. EUROGAMER. *Gradius Collection* [online]. 2006. https://www.eurogamer.net/r-gradiuscollection-psp

[19] MORALES, Raquel. Los mejores juegos shoot'em up de la historia hasta ahora. *Alfa Beta Juega* [online]. [vid. 2022-06-03]. https://alfabetajuega.com/listas/mejores-juegos-shoot-em-up

[20] BIELBY, Matt. The Complete YS Guide to Shoot 'Em Ups. *Your Sinclair Magazine*. 1990, **Issue 55**, 33.

[21] Galaga '90 Review. *GameSpot* [online]. [vid. 2022-06-19]. https://www.gamespot.com/reviews/galaga-90-review/1900-6176451/

[22] *Panzer Dragoon Orta - IGN* [online]. [vid. 2022-06-19]. https://www.ign.com/articles/2003/01/10/panzer-dragoon-orta-2

[23] REED, Kristan. EUROGAMER. *Gyruss* [online]. 2007. https://www.eurogamer.net/r-gyruss-360

[24] *Space Invaders - Videogame by Midway Manufacturing Co.* [online]. [vid. 2022-06-03]. https://www.arcade-museum.com/game_detail.php?game_id=9662

[25] SL, Uptodown Technologies. Radiant (Android). *Uptodown.com* [online]. [vid. 2022-06-03]. https://radiant.uptodown.com/android

[26] *Beat Hazard en Steam* [online]. [vid. 2022-06-03]. https://store.steampowered.com/app/49600/Beat_Hazard/

[27] *Chicken Invaders | InterAction studios* [online]. [vid. 2022-06-03]. http://www.interactionstudios.com/chickeninvaders.php

[28] PÉREZ, Danier Lorenzo. Algoritmos de generación procedural para la creación de mapas en videojuegos 2D de navegador. Trabajo de diploma, 2016.

[29] *COSMOX | Portal de Videojuegos Cubanos* [online]. [vid. 2022-07-03]. https://cosmox.uci.cu/

[30] *COSMOX: nuevo portal de la UCI para videojuegos online* [online]. [vid. 2022-06-19]. https://www.juventudrebelde.cu/ciencia-tecnica/2020-02-20/cosmox-nuevo-portal-de-la-uci-para-videojuegos-online

[31] ORALLO, Enrique Hernández. El Lenguaje Unificado de Modelado (UML), 6.

[32] *Ideal Modeling & Diagramming Tool for Agile Team Collaboration* [online]. [vid. 2022-06-19]. https://www.visual-paradigm.com/

[33] *Unity ¿Qué es y para qué sirve?* | *Tutorial Unity* [online]. [vid. 2022-06-19]. https://www.masterd.es/blog/que-es-unity-3d-tutorial

[34] *Introducción — Blender Manual* [online]. [vid. 2022-06-19]. https://docs.blender.org/manual/es/dev/getting_started/about/introduction.html

[35] GIMP. *GIMP* [online]. [vid. 2022-06-19]. Dostupné z: https://www.gimp.org/

[36] *¿Qué es Gimp y para qué sirve?* [online]. [vid. 2022-06-19]. https://www.seoptimer.com/es/blog/que-es-gimp-y-para-que-sirve/

[37] *Lenguaje de Programación C#* [online]. [vid. 2022-06-19]. http://www.larevistainformatica.com/C1

[38] BILLWAGNER. *A tour of C# - Overview* [online]. [vid. 2022-06-19]. https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[39] LINUX, VISUAL STUDIO CODE Editor de Código Fuente Desarrollador Microsoft Lanzamiento inicial 29 de abril del 2015 Última versión estable 1 29 Plataformas soportadas a Windows Idioma Ingles Licencia Código ABIERTO. *Visual Studio Code - EcuRed* [online]. [vid. 2022-06-19]. https://www.ecured.cu/Visual_Studio_Code

[40] *Documentation for Visual Studio Code* [online]. [vid. 2022-06-19]. https://code.visualstudio.com/docs

[41] PMOINFORMATICA.COM, Publicado por. *Requerimientos no funcionales: Ejemplos* [online]. [vid. 2022-06-20]. http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html

[42] Introducing EdrawMax 10. *Edrawsoft* [online]. [vid. 2022-06-19]. https://www.edrawsoft.com/es/article/package-diagram-uml.html

[43] NUEVA, Leonel Vazquez. Videojuego para la rehabilitación cognitiva enfocado en la percepción. Trabajo de diploma 2018.

[44] DE, Marzo, Carlos REYNOSO a Nicolás KICILLOF. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. 2004.

[45] CRUZ, Alejandro Andrés Pi. Arquitectura de software para videojuegos desarrollados sobre el motor de juego Unity 3D. Trabajo de diploma, 2017.

[46] *PDF de programación - GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO* [online]. [vid. 2022-06-03]. https://www.lawebdelprogramador.com/pdf/9527-GUIA-DE-CONSTRUCCION-DE-SOFTWARE-EN-JAVA-CON-PATRONES-DE-DISENO.html

[47] TABARES, Ricardo Botero. Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*. 2010, **4**(8), 161–173. ISSN 2539-4169.

[48] Unity Documentation. *Unity Documentation* [online]. [vid. 2022-07-03]. https://docs.unity.com/

[49] *Joystick Pack* | *Input Management* | *Unity Asset Store* [online]. [vid. 2022-06-20]. https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631

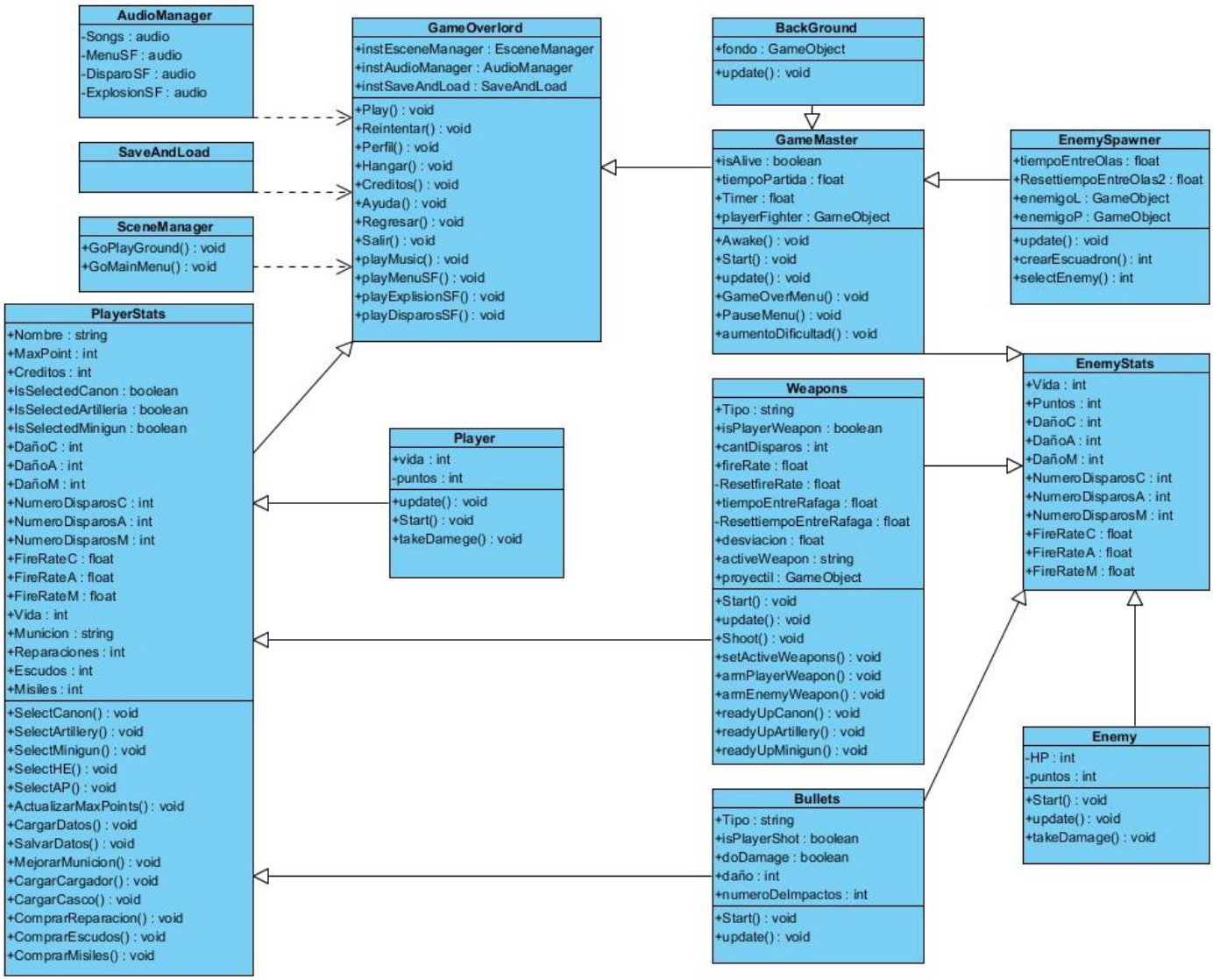
[50] *Kenney • Home* [online]. [vid. 2022-06-20]. https://www.kenney.nl

[51] *Download Free Vector Art, Stock Photos & Stock Video Footage* [online]. [vid. 2022-06-20]. https://www.vecteezy.com/

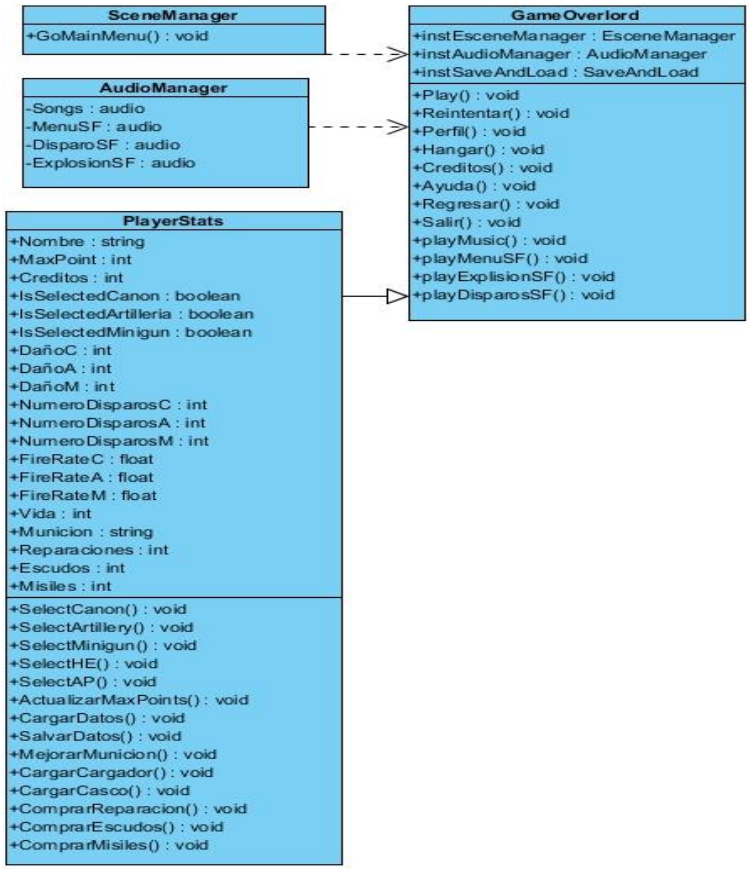
ANEXOS

A.1 Diagramas de clases

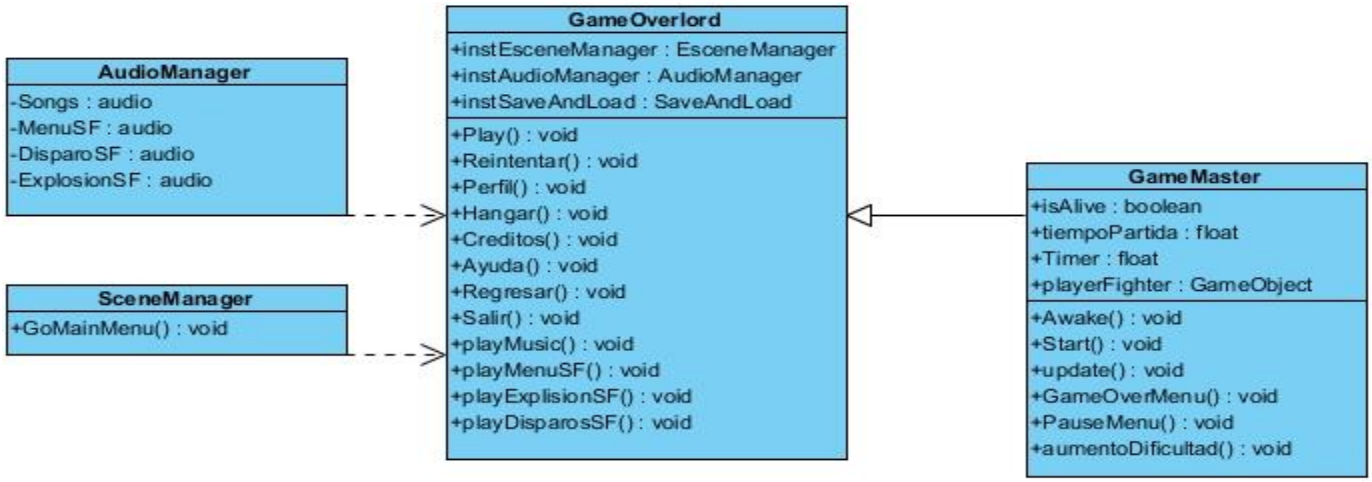
A 1.1 Diagrama de clases del videojuego Ace Fighter



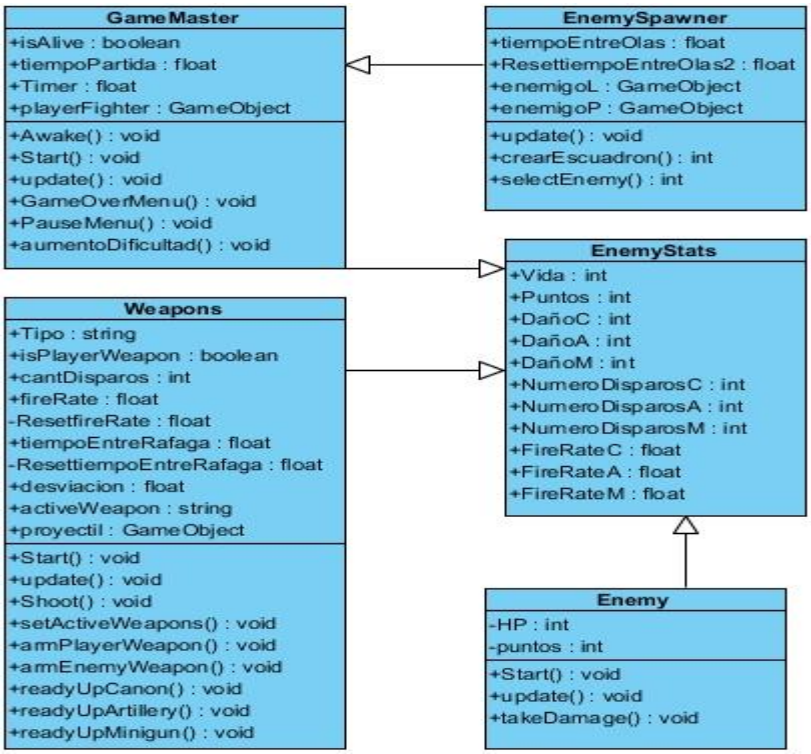
A 1.1 Diagrama de clases del mecanismo M1 Control del menú principal.



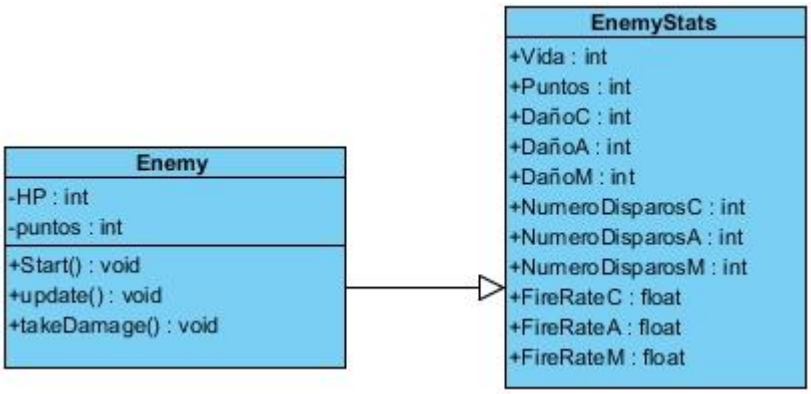
A 1.2 Diagrama de clases del mecanismo M2 Control de partida.



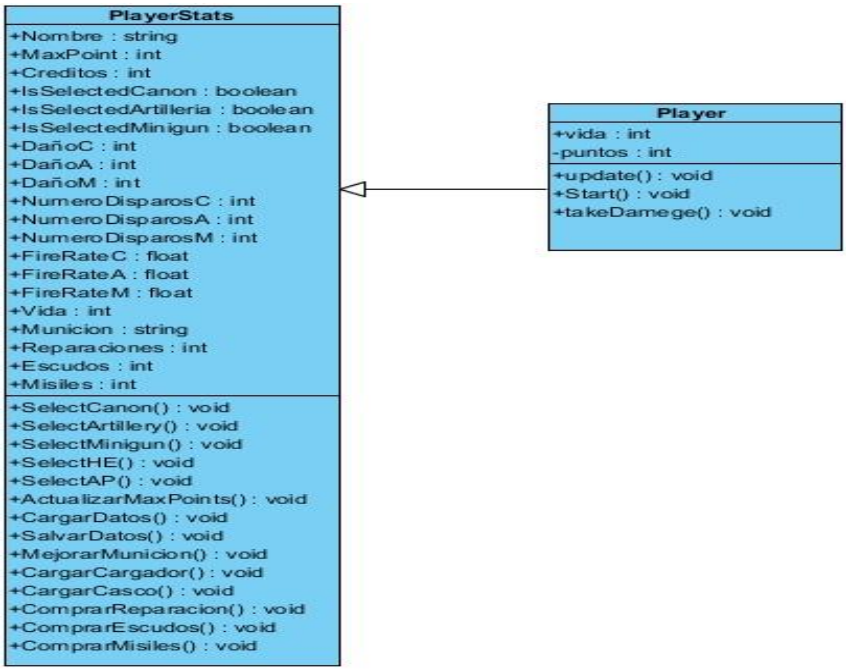
A 1.3 Diagrama de clases del mecanismo M3 Generación de enemigos.



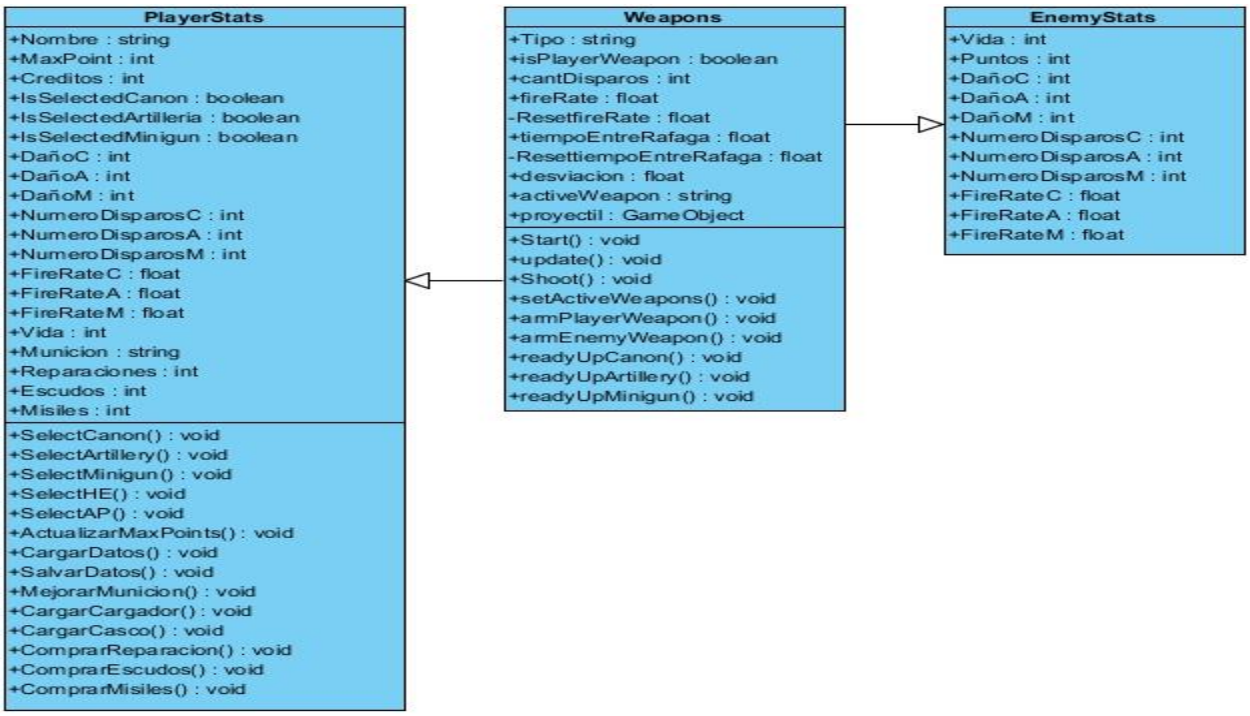
A 1.4 Diagrama de clases del mecanismo M4 Enemigo.



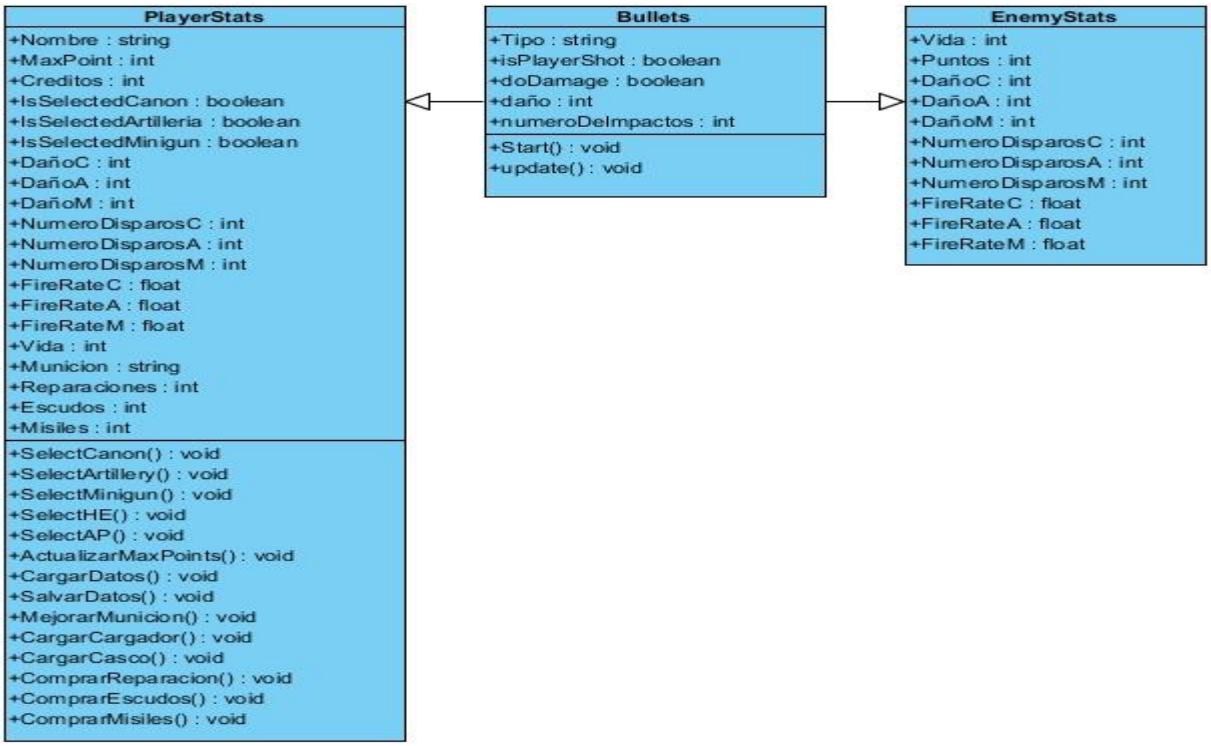
A 1.5 Diagrama de clases del mecanismo M5 Jugador.



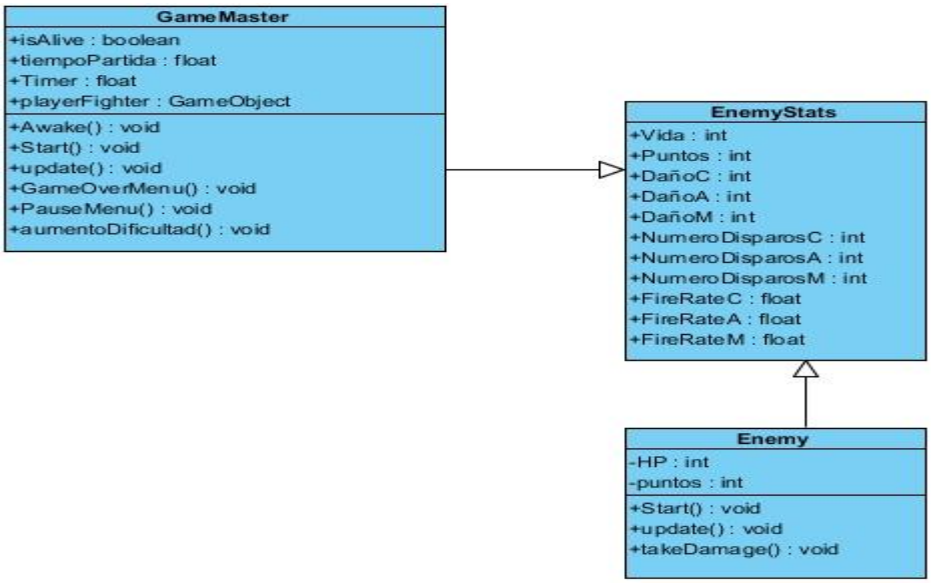
A 1.6 Diagrama de clases del mecanismo M6 Armas.



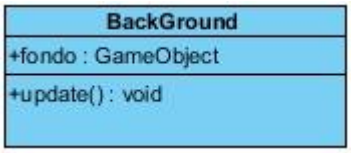
A 1.7 Diagrama de clases del mecanismo M7 Proyectiles.



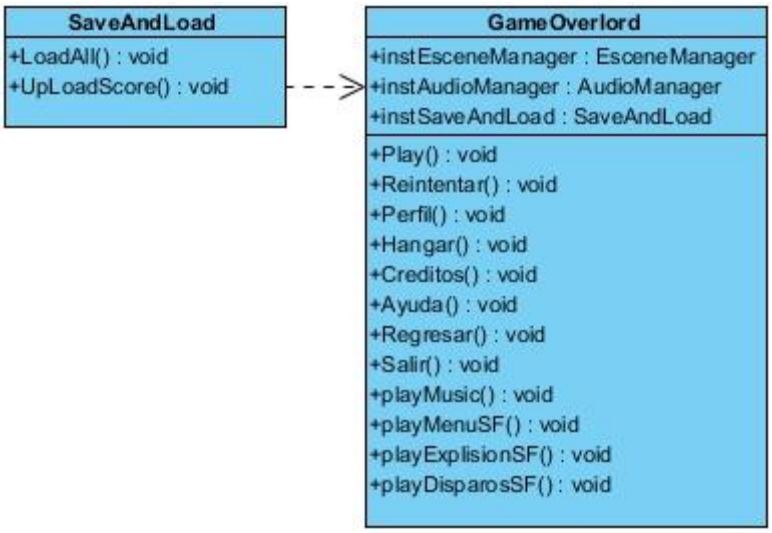
A 1.8 Diagrama de clases del mecanismo M8 Recompensa.



A 1.9 Diagrama de clases del mecanismo M9 Movimiento del escenario.

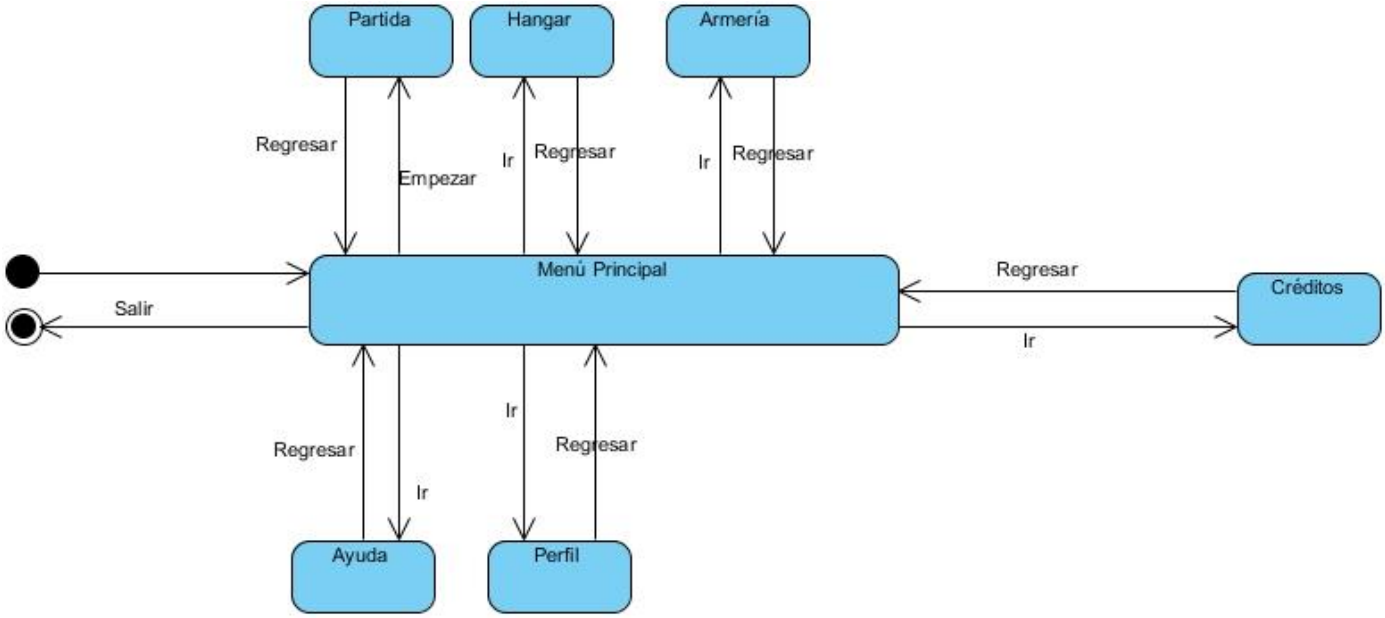


A 1.10 Diagrama de clases del mecanismo M10 Conexión.

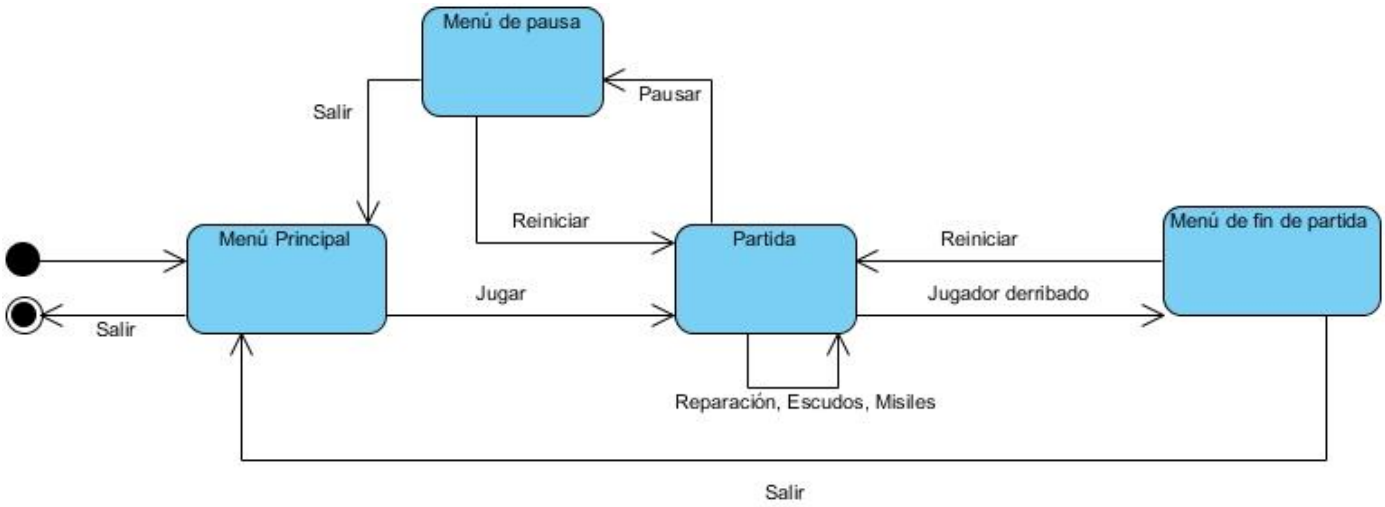


A.2 Diagramas de estado

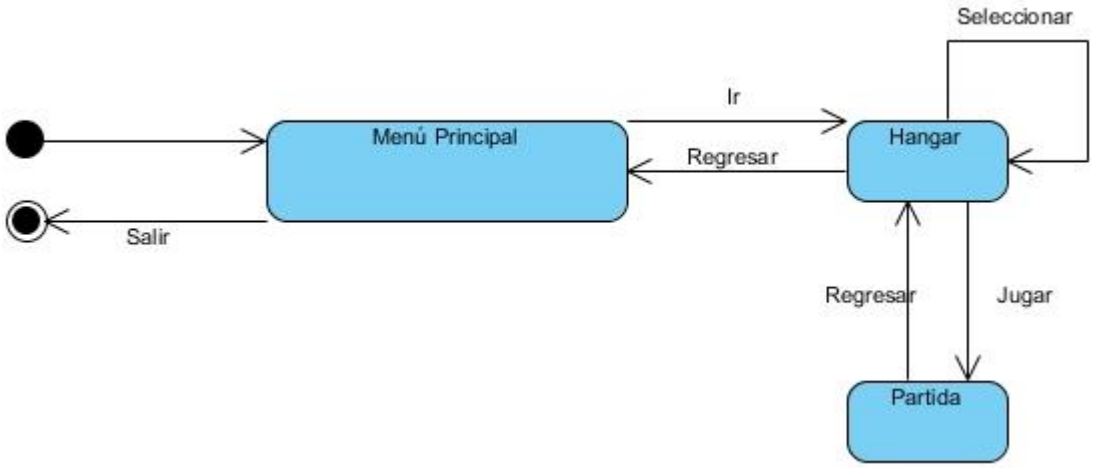
A 2.1 Diagrama de estado del menú principal



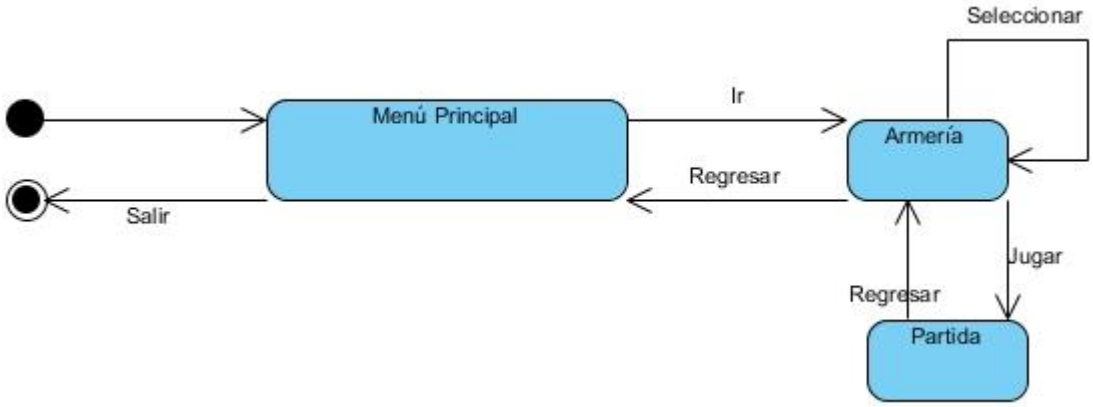
A 2.2 Diagrama de estado en juego



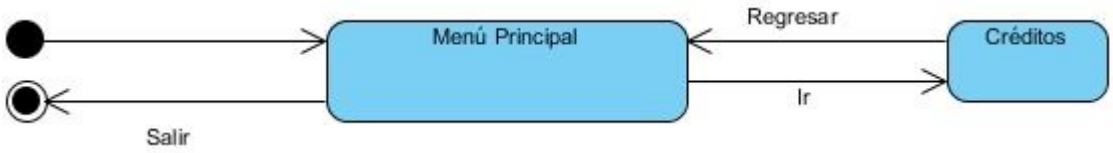
A 2.3 Diagrama de estado hangar



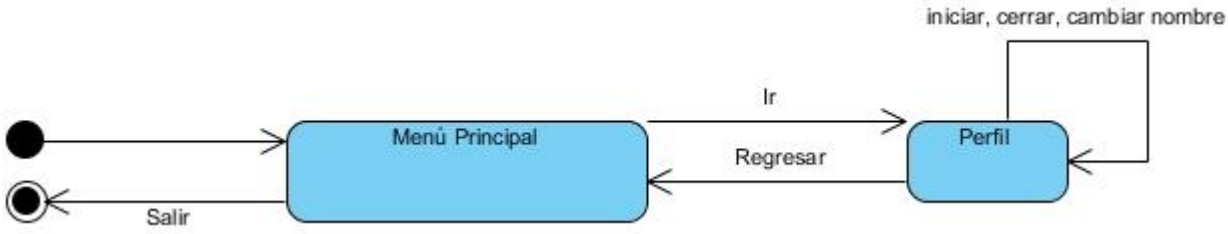
A 2.4 Diagrama de estado armería



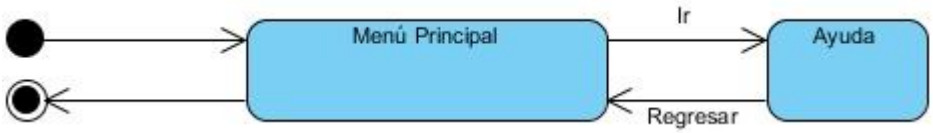
A 2.5 Diagrama de estado créditos



A 2.6 Diagrama de estado perfil

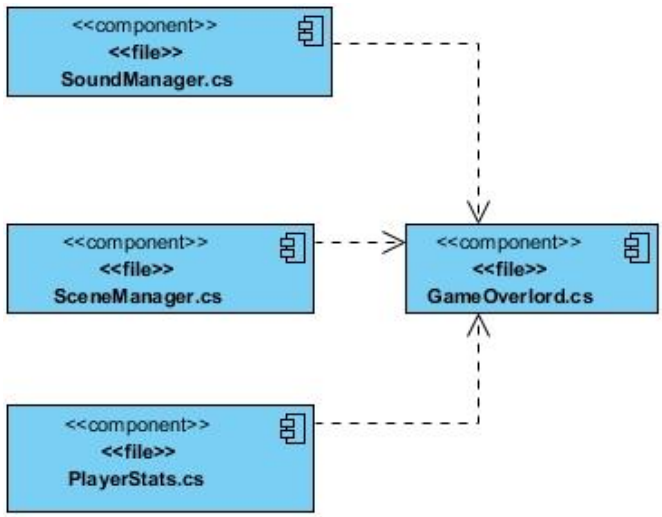


A 2.7 Diagrama de estado ayuda

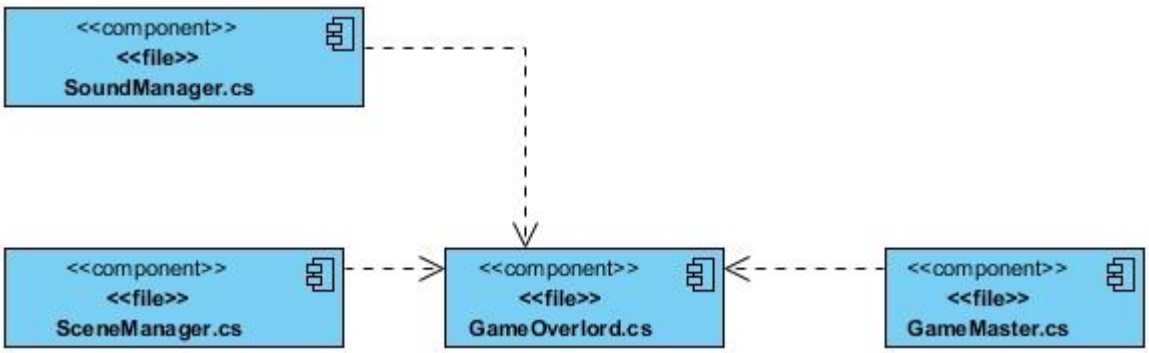


A.3 Diagramas de componentes

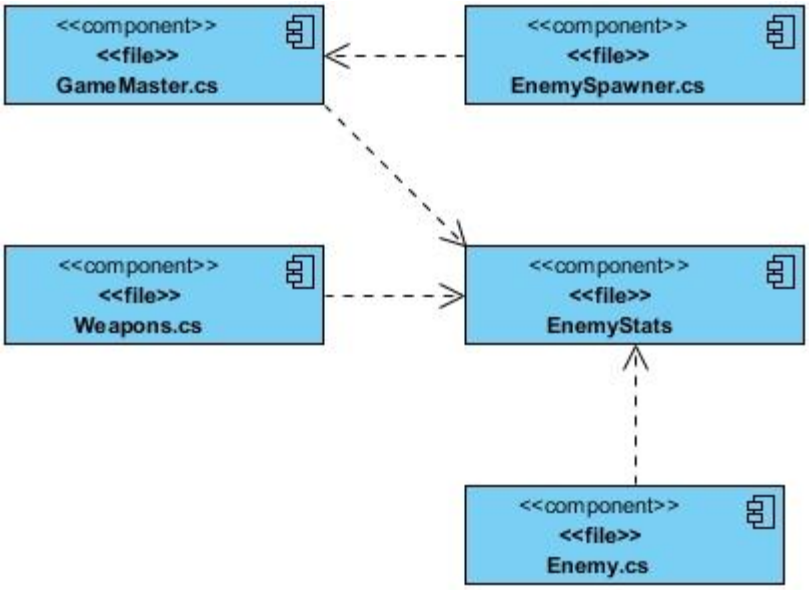
A 3.1 Diagrama de componentes del mecanismo M1 Control del menú principal.



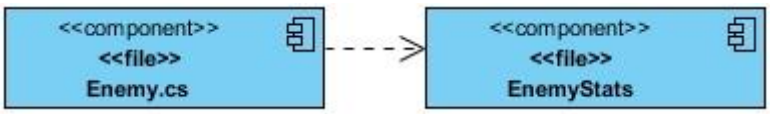
A 3.2 Diagrama de componentes del mecanismo M2 Control de partida.



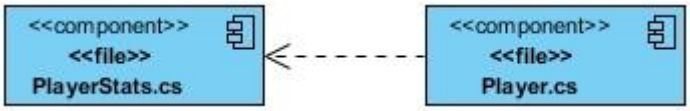
A 3.3 Diagrama de componentes del mecanismo M3 Generación de enemigos.



A 3.4 Diagrama de componentes del mecanismo M4 Enemigo.



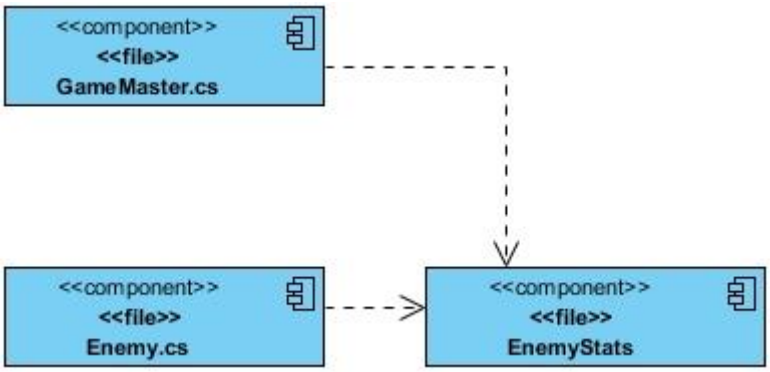
A 3.5 Diagrama de componentes del mecanismo M5 Jugador.



A 3.6 Diagrama de componentes del mecanismo M6 Armas.

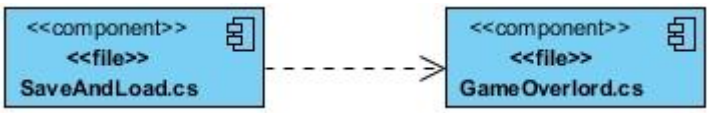
A 3.7 Diagrama de componentes del mecanismo M7 Proyectiles.

A 3.8 Diagrama de componentes del mecanismo M8 Recompensa.



A 3.9 Diagrama de componentes del mecanismo M9 Movimiento del escenario.

A 3.10 Diagrama de componentes del mecanismo M10 Conexión.



A.4 No conformidades registradas durante las pruebas Beta

- Iteración 1:
    - NC 1.1: Error ortográfico.
        - Nombre del probador: Lázaro Ernesto Martínez Gonzales.
        - Localización del error: menú de pausa, la palabra puntuación no está acentuada.
        - Tipo de error: app-Doc/ortografía.
        - Impacto: medio.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.2: Error ortográfico.
        - Nombre del probador: Álvaro Sierra Ariad.
        - Localización del error: menú de fin de partida, la palabra créditos no está acentuada.
        - Tipo de error: app-Doc/ortografía.
        - Impacto: medio.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.3: Cuando se regresa al menú principal se pierden los usos de las habilidades que no fueron consumidos en la partida.
        - Nombre del probador: Mirel Alberto Feria Martínez.
        - Localización del error: menú principal.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.4: El botón reiniciar pone en cero los usos de las habilidades aun cuando se conservan usos de estas.
        - Nombre del probador: Álvaro Sierra Ariad.
        - Localización del error: pantalla de juego.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.

        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.5: El letrero “Derribado” se superpone con la sección de puntuación
        - Nombre del probador: Pedro José Ramírez González.
        - Localización del error: en el menú de fin de partida.
        - Tipo de error: app-error de interfaz.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.6: El texto del botón de selección de artillería se sale demasiado del marco del botón.
        - Nombre del probador: Mirel Alberto Feria Martínez.
        - Localización del error: panel de la armaría.
        - Tipo de error: app-error de interfaz.
        - Impacto: medio.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 1.7: El texto con el nombre de las mejoras se superpone con los botones de mejora y no se pueden leer.
        - Nombre del probador: Pedro José Ramírez González.
        - Localización del error: panel del hangar.
        - Tipo de error: app-error de interfaz.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
- Iteración 2:
    - NC 2.1: El comentario “Serás recordado como un héroe” se superpone con la sección de créditos otorgados.
        - Nombre del probador: Lázaro Ernesto Martínez Gonzales.
        - Localización del error: menú de fin de partida.
        - Tipo de error: app-error de interfaz.

        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.
    - NC 2.2: El botón de la habilidad vida no se activa cuando el jugador cumple los requisitos para usarla.
        - Nombre del probador: Ernesto Alfredo Estrada Casas.
        - Localización del error: panel de habilidades durante la partida.
        - Tipo de error: app-funcionalidad.
        - Impacto: alto.
        - Estado del defecto: resuelto.
        - Respuesta del equipo: resuelto.